

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-139140

(43)Date of publication of application : 20.05.1994

(51)Int.Cl. G06F 12/06
G06F 12/00

(21)Application number : 03-356593

(71)Applicant : INTEL CORP

(22)Date of filing : 25.12.1991

(72)Inventor : ROBINSON KURT B
ELBERT DALE K
LEVY MARKUS A

(30)Priority

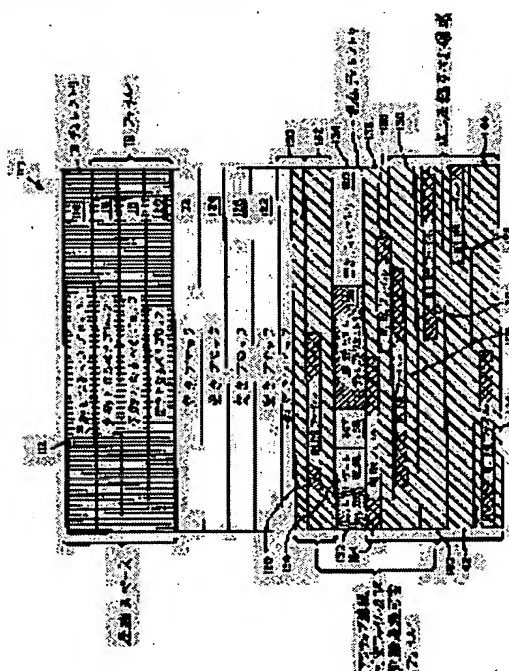
Priority number : 90 636238 Priority date : 31.12.1990 Priority country : US

(54) FILING STRUCTURE OF NONVOLATILE SEMICONDUCTOR MEMORY

(57)Abstract:

PURPOSE: To enable erasure of bit only block as unit and to overwrite a bit from 1st logical state to 2nd logical state without erasing any bit in advance by providing an active block, reserved block and directory block.

CONSTITUTION: While a personal computer system is operated, data or a code is introduced into an empty block, a block 130 is a partially empty block and the active code is contained at one part. Besides, blocks 114, 116 and 118 of a flash memory array 112 are reserved blocks and during a clean-up operation, temporary file backup is executed. Besides, a directory block 134 includes a directory flag 152, boot record 154 and block state table 156 or the like. Then, when the boot record 154 is to be made invalid, a logical value '1' in the total inspection data field of the boot record 154 is overwritten into logical value '0'.



LEGAL STATUS

[Date of request for examination] 26.11.1998

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision
of rejection]

[Date of requesting appeal against examiner's
decision of rejection]

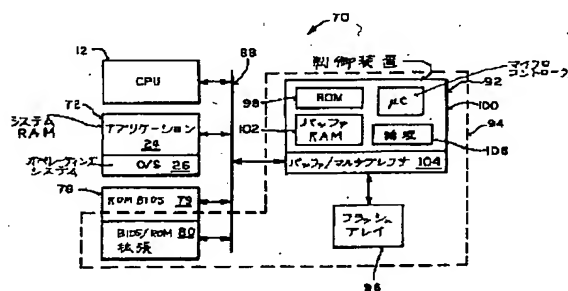
[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(11)特許出願公開番号

(43)公開日 平成6年(1994)5月20日

8526-5B



【特許請求の範囲】

【請求項1】 ブロック単位でのみ消去可能である不揮発性半導体メモリであって、その各ビットをあらかじめ消去せずに第1の論理状態から第2の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第2の論理状態から第1の論理状態に重ね書きすることは可能であるような不揮発性半導体メモリにおいて、

- (A) 第1のファイルを記憶するアクティブブロックと；
- (B) アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第1のファイルのコピーである第2のファイルを記憶する予約ブロックと；
- (C) 第1のファイルを識別するためのディレクトリエントリから成るディレクトリブロックとを具備する不揮発性半導体メモリ。

【請求項2】 (A) ブロック単位で消去可能である不揮発性半導体メモリであって、その各ビットをあらかじめ消去せずに第1の論理状態から第2の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第2の論理状態から第1の論理状態に重ね書きすることは可能であり、

- (1) 第1の複数のファイルを記憶するアクティブブロックと；
- (2) アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第1の複数のファイルのコピーである第2の複数のファイルを記憶する予約ブロックとを含む不揮発性半導体メモリと；
- (B) それぞれが第1の複数のファイルの中の対応する1つのファイルを識別する複数のディレクトリエントリを記憶するメモリとを具備するコンピュータのメモリシステム。

【請求項3】 (A) 第1の複数のファイルを記憶するアクティブブロックと；

- (B) アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第1の複数のファイルのコピーである第2の複数のファイルを記憶する予約ブロックと；
- (C) (1) ディレクトリブロックをディレクトリブロックとして識別するためのフラグと、(2) フラッシュEEPROMのファイル構造に関する情報を含むブート記録と、(3) ブロックが(a) アクティブブロックであるか又は予約ブロックであるか、(b) 未使用であるか又は使用中であるか及び(c) 欠陥を含んでいるか又は含んでいないかに関する情報を提供するブロック状態テーブルと、(4) それぞれが第1の複数ファイルの中の対応する1つのファイルを識別する複数の連係リストディレクトリエントリとを含むディレクトリブロックとを具備するフラッシュ電氣的消去可能プログラマブル読取り専用メモリ(EEPROM)。

【請求項4】 (A) 中央処理装置と；

(B) 各ビットをあらかじめ消去せずに第1の論理状態から第2の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第2の論理状態から第1の論理状態に重ね書きすることは可能であり、

- (1) 第1のファイルを記憶するアクティブブロックと、(2) アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第1のファイルのコピーである第2のファイルを記憶する予約ブロックと、(3) 第1のファイルを識別するためのディレクトリエントリから成るディレクトリブロックを含むブロック単位で消去可能な不揮発性半導体メモリと；
- (C) 不揮発性半導体メモリを制御するコードを記憶する記憶手段とを具備するコンピュータシステム。

【請求項5】 (A) 中央処理装置と；

(B) 各ビットをあらかじめ消去せずに第1の論理状態から第2の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第2の論理状態から第1の論理状態に重ね書きすることは可能であり、

- (1) 第1の複数のファイルを記憶するアクティブブロックと、(2) アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第1の複数のファイルのコピーである第2の複数のファイルを記憶する予約ブロックとを含むブロック単位で消去可能な不揮発性半導体メモリと；
- (C) それぞれが第1の複数のファイルの中の対応する1つのファイルを識別する複数のディレクトリエントリを記憶するメモリと；
- (D) 不揮発性半導体メモリを制御するコードを記憶する記憶手段とを具備するコンピュータシステム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はコンピュータ記憶システムのアーキテクチャの分野に関する。特に、本発明は大形ブロック消去可能不揮発性半導体メモリのファイル記憶システムに関する。

【0002】

【従来の技術】従来のある種のパーソナルコンピュータシステムは、数種類の記憶システムである読取り専用メモリ(「ROM」)、ランダムアクセスメモリ(「RAM」)、大容量記憶のためのハード(すなわち、固定)ディスクドライブ及び出入れ自在磁気フロッピーディスクへの記憶のためのフロッピーディスクドライブに結合するマイクロプロセッサ(中央処理装置ともいう)を含む。フロッピーディスクはディスクettとも呼ばれている。このような従来のパーソナルコンピュータシステムは、通常、それぞれのパーソナルコンピュータシステムの一部を形成する記憶システムに特別に適應するアーキテクチャを有する。

【0003】ROMと、ROMに記憶されているROM

モニターと呼ばれるプログラムとを併せてファームウェアという。ROMベーシング入出力システム(BIOS)モジュールはROMに記憶されて、ある種のパーソナルコンピュータについてオペレーティングシステムにより使用されるROMモニターの1例である。ROM BIOSモジュールは、通常、(1) キーボード、ディスクドライブ及びプリンタを含めたいくつかのハードウェアのドライバと；(2) パワーオン自己試験プログラム(「POST」)と；(3) システムを初期設定するスタートアップルーチンと；(4) ディスケット又はハードディスクからブート、すなわち第1のセクターを読取るローダプログラムとを含む。

【0004】パーソナルコンピュータがオンされた後、BIOSモジュールのPOSTプログラムが実行され、BIOSスタートアップルーチンはいくつかの初期設定を実行し、ローダプログラムはディスク又はハードディスクからブートセクターの内容を読取る。そのブートセクターはコンピュータのオペレーティングシステムのローダプログラムを含む。オペレーティングシステムのローダプログラム自体は、ディスク又はハードディスクからRAMへオペレーティングシステムの一部をロードする。

【0005】パーソナルコンピュータのオペレーティングシステムは指令を処理し、プログラムの実行を制御し、コンピュータシステムのハードウェア資源及びソフトウェア資源を監視する。従来のオペレーティングシステムの1つの型がワシントン州レッドモンドのマイクロソフト・コーポレーションから販売されているMS-DOSである。MS-DOSオペレーティングシステムは先に述べたローダプログラムと、MS-DOS BIOSと、MS-DOSカーネルと、ユーザーインタフェースと、ユーティリティプログラムとを含む。

【0006】MS-DOS BIOSは(1) ROM BIOSのドライバの構成要素を拡張し且つ使用する新たなドライバと；(2) MS-DOS BIOSドライバの初期設定ルーチンと；(3) 別のローダプログラムとを含む。

【0007】MS-DOS BIOSの新たなドライバをBIOS拡張又はBIOS ROM拡張ともいう。従来のパーソナルコンピュータの中には、ROMの32キロバイト部分又は64キロバイト部分及びそれより小さい(通常は2キロバイトの)BIOS ROM拡張に記憶されているソフトウェアによって全ての入出力機能を駆動するものがある。

【0008】MS-DOS BIOSドライバの初期設定ルーチンはコピーライト公報メッセージを表示すると共に、新たなドライバに関する割込みテーブルを調整する。MS-DOS BIOSのローダプログラムはオペレーティングシステムの残る部分をロードする。

【0009】MS-DOSカーネルはBIOSとアプリ

ケーションプログラムとの間のシェルである。MS-DOSカーネルはアプリケーションプログラムの実行を開始させ、アプリケーションプログラムのためのメモリを割付け、アプリケーションプログラムとハードウェアとの間にアプリケーションプログラムインタフェースを構成し且つファイルの読取りと書込みを管理する。

【0010】MS-DOSのユーザーインタフェースはユーザーに情報を提供する。ユーザーインタフェースは、ユーザーに指令の入力を促すプロンプトを与える。オペレーティングシステムが制御中であるとき、ユーザーインタフェースはシステムのマネージャとして動作する。

【0011】MS-DOSのユーティリティプログラムはMS-DOSに関するいくつかの有用な機能を提供する。それらの機能には(1) ディスケット又はハードディスクの定様式化及び(2) ディスケット又はハードディスクの検査がある。

【0012】先に述べたようなパーソナルコンピュータシステムと共に使用されるハードディスクやディスクは不揮発性記憶システムである。すなわち、コンピュータへの給電が停止したときにデータは失われないのである。ハードディスク及びディスクはブロック記憶装置の1種である。すなわち、データはブロック単位で転送される。

【0013】ハードディスクやディスクの場合、データは同心のトラックに物理的に記憶される。それぞれのトラックは複数のセクターから構成されている。通常、セクターは512バイトの長さで固定されている。パーソナルコンピュータシステムのディスク制御装置と物理装置ドライバは、常にディスクのセクターエントリとの間でデータの書込み及び読取りを実行するのが普通である。

【0014】クラスタは論理的にアドレッシング可能な最小の記憶単位である。ハードディスクによっては、クラスタごとに4つのセクターを含む。他のハードディスクや高密度3.5インチディスクの場合には、1つのクラスタ単一のセクターである。

【0015】ハードディスクの各区画領域は、その独自のオペレーティングシステムを記憶することができる論理サブシステムを形成する。定様式化ハードディスクの第1のセクターにある区画テーブルは、区画に関する情報を含む。

【0016】ハードディスクとディスクは、最初に使用される前に定様式化される。低レベルの定様式化は各トラックを複数のセクターに分割し、識別(「ID」)セクターヘッダをトラックに沿って均一に配分された位置に配置する。高レベル定様式化はクラスタを確定し、いくつかのディスク領域を初期設定し、データを受信するためにディスクの準備を整える。

【0017】装置ドライバ及びBIOSレベルでは、デ

ィスク要求は、ドライブ、ヘッド、シリンダ／トラック、セクター及び長さを指示する「チューブル」により記述される。DOSレベル及びBIOSレベルでは、論理セクター番号はわかっている。DOSはディスクのクラスタエントリとの間で読取り及び書込みを実行する。

【0018】図1は、従来のMS-DOSオペレーティングシステムの論理的編成を示す。MS-DOSオペレーティングシステムの場合、ディスク2はシステム領域4と、データ領域9という2つの論理領域に分割されている。システム領域はブートレコード3と、ファイル割付けテーブル（「FAT」）5と、ルートディレクトリエントリを含むルートディレクトリ7とを含む。データ領域9は、アプリケーションプログラム、データ及びサブディレクトリ情報を記憶するために使用されるファイルを含む。

【0019】ブートレコード3は、オペレーティングシステムをロードするブートストラップロードプログラムを含む。ブートレコード3は、定様式化DOSのASCII名と、ディスクのセクターごとのバイト数と、クラスタごとのセクター数と、ブートレコード中のセクター数と、ファイル割付けテーブルのコピーの数と、ルートディレクトリエントリの数と、区画ごとのセクターの数と、ディスク型番号と、ファイル割付けテーブルのエントリ（すなわち、クラスタ）ごとのセクターの数と、トラックごとのセクター数と、ディスクごとの面の数と、予約セクター又はかげのセクターの数と、物理ドライブ番号と、拡張ブートセクターシグナチャと、ボリューム識別と、ボリュームテーブルとに関する情報をさらに含む。

【0020】ルートディレクトリ7は、それぞれがファイルの何らかの属性を記述する複数の32バイトエントリから成るテーブルである。通常、ルートディレクトリ7を構成する各ディレクトリエントリはファイル名と、ファイル拡張と、属性フラグと、ファイルの日付スタンプと、ファイルを構成するクラスタの開始クラスタ番号と、ファイルサイズとを含む。

【0021】ディスクの各ファイルは1つ又は複数のクラスタから構成されている。ファイル割付けテーブル5は、ファイルを構成するクラスタが互いに連係する連鎖の形態をとる記録を含む。典型的なFAT5はクラスタごとに1つずつある複数の2バイトエントリから成るリストを含む。従来のFATの中には、FATエントリが2バイトを越えるものもある。各FATエントリの長さはクラスタの総数によって決まる。ファイルのディレクトリエントリはそのファイルの開始クラスタ番号を含み、オペレーティングシステムはその開始クラスタ番号を使用して、ファイル割付けテーブルをアクセスする。各FATエントリはファイルの次のクラスタを指示するポインタである。従って、その最初のアクセスによって検索されるFATエントリは、ファイルを構成している

次のクラスタのクラスタ番号を含む。オペレーティングシステムはその次のクラスタ番号を使用し、FATをアクセスしてさらにその次のクラスタ番号を検索し、FAT5の中の特別のマーカに到達するまでこのプロセスを継続してゆく。

【0022】ディスクにおけるファイル構造はツリー形である。ルートディレクトリのエントリはサブディレクトリを指示するポインタを形成することができる。サブディレクトリは入れ子形であっても良い。

【0023】ある種の従来のパーソナルコンピュータでハードディスクやディスクセットを使用することに関連する欠点の1つは、ハードディスクドライブ及びフロッピーディスクドライブが多数の機械的構成要素を伴う相対的に物理的に大形の装置であるという点である。その大きさは、パーソナルコンピュータの他の多くの部品を構成している集積回路の小ささと対照的である。さらに、相対的に大形である従来の典型的なハードディスクドライブやフロッピーディスクドライブは、持ち運び自在のパーソナルコンピュータをより一層小型にし、持ち運びしやすくしようとする試みには妨げとなる。

【0024】従来のハードディスクドライブ及びフロッピーディスクドライブに関わるもう1つの欠点は、それらの装置がパーソナルコンピュータの他の部品を構成している集積回路と比較して大量の電力を消費することである。

【0025】従来のハードディスクドライブ及びフロッピーディスクドライブに関わるさらに別の欠点は、それらの装置が過剰な衝撃や振動又はちり、もしくは他の大気中汚染物質にさらされた場合に故障しやすいことである。

【0026】従来の別の種類の不揮発性コンピュータメモリはフラッシュ電氣的消去可能プログラマブル読取り専用メモリ（「フラッシュEEPROM」）である。フラッシュEEPROMはユーザー側でプログラムすることができ、一度プログラムしてしまえば、消去されるまでデータを保持する。フラッシュEEPROMを電氣的に消去すると、装置のメモリの内容全てが1回の相対的に速い動作で消去される。その後、フラッシュEEPROMを新たなコードでプログラムすれば良い。

【0027】ところが、従来のある型のフラッシュEEPROMには、個々のビットセルをあらかじめ消去せず論理値0から論理値1に重ね書きすることは不可能であるという欠点がある。従来の1つの型のフラッシュEEPROMのもう1つの欠点は、大きなブロックの単位で又は装置全体を消去してしまうように消去しなければならないこと、すなわち、論理状態1にリセットしなければならないことである。

【0028】従来の1つの型のフラッシュEEPROMの別の欠点は、フラッシュEEPROMが故障するまでのフラッシュEEPROMの消去サイクルと書込みサイ

クルの数に限りがあることである。

【0029】従来のいくつかのフラッシュEEPROMに関わる重ね書き及び消去についての制限は、場合によっては、パーソナルコンピュータシステムにおけるフラッシュEEPROMの有用性を限定していた。

【0030】

【発明が解決しようとする課題】本発明の目的の1つは、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第1の論理状態から第2の論理状態に重ね書きすることは不可能である不揮発性半導体メモリのファイル構造を提供することである。

【0031】本発明の別の目的は、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第1の論理状態から第2の論理状態に重ね書きすることは不可能であり、ファイル構造の保全性及び信頼性を最大にするのを助長する予約ブロックを含むような不揮発性メモリのファイル構造を提供することである。

【0032】本発明の別の目的は、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第1の論理状態から第2の論理状態に重ね書きすることは不可能であり、アクティブブロックと、予約ブロックと、ディレクトリブロックとを具備する不揮発性半導体メモリを含むコンピュータシステムを提供することである。

【0033】

【課題を解決するための手段】ブロック単位でのみ消去可能である不揮発性半導体メモリを説明する。不揮発性半導体メモリの各ビットをあらかじめ消去せずに第1の論理状態から第2の論理状態に重ね書きすることは不可能である。不揮発性半導体メモリの各ビットをあらかじめ消去せずに第2の論理状態から第1の論理状態に重ね書きすることは可能である。不揮発性半導体メモリは、第1のファイルを記憶するアクティブブロックと、第2のファイルを記憶する予約ブロックと、ディレクトリブロックとを具備する。第2のファイルは第1のファイルのコピーである。コピーはアクティブブロックの消去に先立ってクリーンアップ動作中に作成される。ディレクトリブロックは、第1のファイルを識別するためのディレクトリエントリを含む。

【0034】本発明のその他の目的は、特徴及び利点は添付の図面及び以下の詳細な説明から明らかになるであろう。

【0035】本発明を添付の図面の各図に例示してあるが、図面は本発明を限定するものではない。また、図中、同じ図中符号は同様の要素を指示する。

【0036】

【実施例】図2は、1つの好ましいファイルシステム（ファイル構造ともいう）32を含むパーソナルコンピュータシステム10を示す。ファイルシステム32は1つのフラッシュEEPROM又はフラッシュメモリアレイ34を形成する複数のフラッシュEEPROMを含

む。フラッシュEEPROMは半導体メモリの種類である。ファイル構造32は、システムRAM22に記憶されているソフトウェアファイルシステムドライバ28をさらに含む。パーソナルコンピュータシステム10は中央処理装置（「CPU」12）と、バス18とをさらに含む。

【0037】以下にさらに詳細に説明する通り、フラッシュメモリアレイ34は動的メモリ再割当てを可能にするような構造である。フラッシュメモリアレイ34は可変メモリ構造又はセクター方式ファイル構造のいずれかを有する。フラッシュメモリアレイ34と関連して連係リストディレクトリ又はディスクアナログディレクトリのいずれかを使用する。ディスクアナログディレクトリはバイト消去可能EEPROMなどの個別のメモリに常駐している。連係リストディレクトリはフラッシュメモリ、又はバイト消去可能EEPROMなどの別個のメモリに常駐している。

【0038】システムRAM22はアプリケーションプログラム24を記憶するためのスペースを含む。本発明の一実施例では、システムRAM22は1メガバイトの大きさである。システムRAM22は、他のアプリケーションプログラムを記憶するためと、データのためのスペースをさらに含む。本発明の一実施例においては、システムRAM22はコンピュータシステム10に対して1つのオペレーティングシステム26を含む。別の実施例では、システムRAM22はさらに2つ以上のオペレーティングシステムを含む。

【0039】システムRAM22はファイルシステムドライバソフトウェア28をさらに含む。ファイルシステムドライバソフトウェア28はフラッシュメモリアレイ34のファイル構造を規定する。パーソナルコンピュータシステム10では、中央処理装置12は、フラッシュメモリアレイ34に関わるあらゆるファイルシステム管理ユーティリティを処理するために、ファイルシステムドライバソフトウェア28のフォアグラウンドタスクソフトウェアルーチンを実行する。ファイルシステムドライバソフトウェア28と、フラッシュメモリアレイのファイル構造については以下にさらに詳細に論じる。

【0040】システムRAM22がアプリケーションプログラム24、オペレーティングシステム26及びファイルシステムドライバソフトウェア28を記憶するのみならず、アプリケーションプログラム24、オペレーティングシステム26及びファイルシステムドライバソフトウェア28は、パーソナルコンピュータシステム10がオフ又はオンである間に、フラッシュメモリアレイ34にも記憶される。フラッシュメモリアレイ34は2つ以上のアプリケーションプログラムと、2つ以上のオペレーティングシステムと、2つ以上のファイルシステムドライバソフトウェアとを記憶できる。フラッシュメモリアレイ34はデータファイルと、ディレクトリ情報と

をさらに記憶する。ウォームブート又はコールドブートのいずれかの時点で、ファイルシステムドライバソフトウェア28はフラッシュメモリアレイ34からシステムRAM22へロードされる。要するに、フラッシュメモリアレイはハードディスクの代わりとなる大容量記憶装置として動作する。

【0041】パーソナルコンピュータシステム10は、自身に関わるBIOSソフトウェアを記憶するROM14をさらに含む。本発明の一実施例では、ROM14に記憶されるBIOSは、それぞれ2キロバイトの追加BIOS ROM拡張部を伴う64キロバイトBIOSを含む。ファイルシステムドライバソフトウェア28は、実行に先立って、ROM14ではなくフラッシュメモリアレイ34に記憶される。

【0042】コンピュータシステム10のパワーアップ後、ROM14のBIOSプログラムを実行する時点で、ファイルシステムドライバソフトウェア28とオペレーティングシステム26はフラッシュメモリアレイ34からバス18を介してシステムRAM22へロードされる。そこで、コンピュータシステム10のユーザーはオペレーティングシステム26を使用して、アプリケーションプログラム24をシステムRAM22にロードする。コンピュータシステム10のユーザーは、さらに、データ、他のアプリケーションプログラム及び他のオペレーティングシステムをシステムRAM22にロードすることができる。

【0043】本発明の一実施例では、フラッシュメモリアレイ34はブロックごとに消去され、各ブロックは多数のビットから構成されている。別の実施例においては、フラッシュメモリアレイ34はことごとく消去される。

【0044】フラッシュメモリは、消去に関して、従来の電氣的消去可能プログラマブル読取り専用メモリ

(「EEPROM」)とは異なる。従来のEEPROMは個々のバイト消去制御のためにセレクトトランジスタを使用する。それに対し、フラッシュメモリは個々のトランジスタセルによってはるかに高い密度を達成する。消去モードの間に、ブロックの中のあらゆるメモリセル又はチップ全体にあるあらゆるメモリセルのソースに同時に高電圧を供給すると、その結果、アレイ全体又はブロック全体が消去されることになる。

【0045】フラッシュメモリアレイ34の場合、論理値「1」は、1つのビットセルと関連しているフローティングゲートに、あったとしてもごくわずかの電子しか記憶されていないということを意味する。論理値「0」は、ビットセルと関連するフローティングゲートに多数の電子が記憶されていることを意味する。フラッシュメモリアレイ23が消去された後、フラッシュメモリアレイ23の各ビットセルには論理値1が記憶されている。フラッシュメモリアレイ23の個々のビットセルを、あ

らかじめ消去せずに、論理値0から論理値1に重ね書きすることは不可能である。ただし、フラッシュメモリアレイ23の個々のビットセルを論理値1から論理値0に重ね書きすることは、それが、単に、本来、消去状態と関連する数の電子を含んでいるフローティングゲートに電子を追加することのみを伴うのであれば可能である。

【0046】フラッシュメモリアレイ34は(1) 一度に1ビットずつ、(2) 一度に1語ずつ又は(3) 一度に語群単位のいずれかの方法でプログラム可能である。1つの語は単一のメモリシステムアドレス又は単一の装置と関連する複数のメモリビットから構成される。一度にプログラムされる語群はフラッシュメモリアレイ34の1つの消去ブロックと同じ大きさであることができる。フラッシュメモリに関するプログラム動作を書込み動作ともいう。

【0047】先の述べた通り、フラッシュメモリアレイ34の各ビットを、あらかじめ消去せずに、論理状態0から論理状態1に重ね書きすることは不可能である。論理値0から論理値1への重ね書きに先立ってこのように消去が必要であるために、フラッシュメモリと関連して1つの機能動作が導入される。

【0048】フラッシュメモリアレイ34は実行可能コードと、実行不能データの双方を記憶する。以下の詳細な説明の中では、総称用語としての「データ」は(1) 実行不能データのみを表わすが、(2) 実行可能コードと実行不能データの双方を表わすために使用される。

【0049】フラッシュメモリアレイ34に関連する読取り動作は、他の読取り専用メモリ装置に関連する読取り動作とごく類似している。一実施例では、フラッシュメモリアレイ34の読取り動作は135ナノ秒を要する。ところが、フラッシュメモリアレイの書込み動作と消去動作はそれよりはるかに遅い。本発明の一実施例においては、フラッシュメモリアレイ34の1つのブロックの消去時間は1秒である。フラッシュメモリアレイ34の1つの語の書込み動作は10マイクロ秒を要する。従って、フラッシュメモリアレイ34に関連する読取り動作と、書込み・消去動作とは非対称の関係にあり、書込み動作と消去動作は読取り動作より著しく遅い。

【0050】さらに、フラッシュメモリアレイ34の各フラッシュメモリは消去並びに書込み(すなわち、プログラム)動作に関して限られた耐久寿命を有する。たとえば、一実施例では、フラッシュメモリアレイ34は、故障し始めるか又は消去/書込み性能の劣化を示すまでに少なくとも10000消去/プログラムサイクルに耐えることができる。従って、本発明の一実施例においては、フラッシュメモリアレイ34が完了できる消去/書込みサイクルの数に限りがある。これに対し、フラッシュメモリアレイ34が読取りを実行できる回数については、そのような最長寿命は存在していない。

【0051】本発明の一実施例では、フラッシュメモリアレイ34を構成している各フラッシュメモリは、それぞれのメモリ装置の中に複数の消去ブロックを有している。その実施例の場合、隣接するブロックを巡る消去／プログラミングの循環には制限がある。隣接ブロックとは、共通の行接続又は共通の列接続を共有しているブロックである。隣接ブロックを巡る循環の制限は、電気的データ妨害状態を阻止するために設けられる。

【0052】循環の頻度の多いブロックをホットブロックといい、循環の起こらないか又は循環の頻度が少ないブロックをコールドブロックという。重要なパラメータは、隣接ブロックとそれぞれ関連するサイクルの数の差である。ホットブロックは、最終的には、フラッシュメモリアレイ34のコールドブロックではデータが悪影響を受けてしまうようなサイクルカウントに達することもある。このホットブロック／コールドブロック相互作用を回避するための1つの方法は、フラッシュメモリアレイ34のホットブロックにサイクルの制限を課すことである。ホットブロック／コールドブロック相互作用の悪影響を最小限に抑える別の方法は、フラッシュメモリアレイ34のコールドブロックを周期的に消去し、再プログラミングするというものである。コールドブロックの周期的な消去と再プログラミングをコールドブロックの再生という。

【0053】以下にさらに詳細に説明する通り、本発明の好ましいファイル構造は、フラッシュメモリアレイ34を構成するフラッシュEEPROMの先に挙げたような機能上の特性を考慮に入れている。

【0054】図3は、1つの好ましいファイル構造58を有するパーソナルコンピュータシステム40を示す。パーソナルコンピュータシステム40は中央処理装置12と、ROM42とを含む。ROM42はBIOSソフトウェア43と、BIOS拡張ソフトウェア44とを記憶している。パーソナルコンピュータシステム40はバス48と、システムRAM52とをさらに含む。システムRAM52はアプリケーションプログラム24と、オペレーティングシステム26とを記憶している。パーソナルコンピュータシステム40はRAMバッファ62と、フラッシュメモリアレイ64とをさらに含む。

【0055】RAMバッファ62はフラッシュメモリアレイ64とシステムRAM52との間のファイルの転送に関して緩衝を実行する。さらに、RAMバッファ62はフラッシュメモリアレイ64に記憶されているファイルについてのディレクトリの構成又は更新に関しても緩衝を実行する。

【0056】1つ又は複数のフラッシュEEPROMがフラッシュメモリアレイ64を構成している。フラッシュメモリアレイ64はアプリケーションプログラム24と、オペレーティングシステム26と、データと、他のアプリケーションプログラムと、他のオペレーティング

システムを記憶する。オペレーティングシステム26、アプリケーションプログラム24及び他のプログラムとデータはフラッシュメモリアレイ64からバス48を介してシステムRAM52へ転送可能である。

【0057】ファイル構造58はフラッシュメモリアレイ64と、RAMバッファ62と、ROM BIOS拡張ソフトウェア44と、ROM BIOSソフトウェア43の一部とを含む。

【0058】フラッシュメモリアレイ64のファイル構造を規定するファイルシステムドライバソフトウェアはROM42にあるROM BIOSソフトウェア43及びROM BIOS拡張ソフトウェア44の一部を形成している。図3に示す実施例では、ファイルシステムドライバはパーソナルコンピュータシステム40の一部であり且つそれと一体である。BIOSソフトウェア43とBIOS拡張ソフトウェア44は、直接マッピング方式又はページング方式のメモリブロックである。従って、図3に示す実施例においては、ファイルシステムドライバソフトウェアはシステムRAM52にも、フラッシュメモリアレイ64にも記憶されない。

【0059】図4は、さらに別の好ましいファイル構造94を有するパーソナルコンピュータシステム70を示す。パーソナルコンピュータシステム70は中央処理装置12と、システムRAM72とを含む。好ましい実施例では、システムRAM72はアプリケーションプログラム24と、オペレーティングシステム26とを記憶している。パーソナルコンピュータシステム70はバス88と、ROM78とをさらに含む。ROM78はBIOSソフトウェア79と、BIOS拡張ソフトウェア80とを記憶している。

【0060】パーソナルコンピュータシステム70は制御装置92と、フラッシュメモリアレイ96とをさらに含む。フラッシュメモリアレイ96は複数のフラッシュEEPROMから構成されている。フラッシュメモリアレイ96はアプリケーションプログラム24と、オペレーティングシステム26と、データと、他のアプリケーションプログラムと、他のオペレーティングシステムとを記憶し、それらをバス88を介してシステムRAM72にロードすることができる。ファイル構造94は制御装置92と、フラッシュメモリアレイ96と、ROM BIOS拡張ソフトウェア80と、ROM BIOSソフトウェア79の一部とから構成されている。

【0061】図4に示すパーソナルコンピュータシステム70は、フラッシュメモリアレイ96に関して完全ハードウェア駆動形ファイル構造を有する。制御装置92は、フラッシュメモリアレイ96のファイル構造を規定し且つ制御するための特定専用装置である。制御装置92は、制御を監督するマイクロコントローラ100を含む。

【0062】制御装置92は制御論理106をさらに含

む。本発明の好ましい一実施例においては、制御論理106はプログラマブル論理アレイ(「PLA」)から構成される。制御論理106は、制御装置92の動作を制御するデジタル論理を含む回路である。

【0063】一実施例では、制御装置92は、ファイルシステムドライバソフトウェアを記憶するROM98を含む。ファイルシステムドライバソフトウェアはマイクロコントローラ100により実行可能であり、フラッシュメモリアレイ96のファイル構造を規定する。

【0064】別の実施例によれば、フラッシュメモリアレイ96のファイル構造を規定する実行可能なファイルシステムドライバソフトウェアは、ROM BIOSソフトウェア79の一部及びROM BIOS拡張ソフトウェア80を形成する。

【0065】制御装置92はバッファRAM102を含む。バッファRAM102はファイルの転送と、ディレクトリの作成及び更新とに関して緩衝を実行する。

【0066】制御装置92はバッファ/マルチプレクサ104をさらに含む。バッファ/マルチプレクサ104はファイルの転送に関して緩衝を実行する。バッファ/マルチプレクサ104は、さらに、フラッシュメモリアレイ96に出入りするデータを多重化する。

【0067】論理バッファ/マルチプレクサ回路104と関連する論理はいずれかの標準システムバスインタフェースに準拠していれば良い。たとえば、バッファ/マルチプレクサ104と関連する論理はPC XT, PC

AT(すなわち、IDE-工業規格アーキテクチャ)、EISA(すなわち、拡張工業規格アーキテクチャ)、MCA(すなわち、マイクロチャネルアーキテクチャ)、VME(すなわち、仮想機械環境)及びマルチバスなどの規格のうち1つ又は2つ以上に準拠していれば良い。

【0068】本発明の一実施例においては、制御装置92はパーソナルコンピュータシステム70の中に含まれている。たとえば、制御装置92はパーソナルコンピュータシステム70内部のシステムボード(図示せず)に位置していることも可能であろう。

【0069】別の実施例では、制御装置92は、パーソナルコンピュータシステム70の外にある又はパーソナルコンピュータシステム70の拡張用スロット(図示せず)の中にある入出力(「I/O」)装置であっても良い。その実施例の場合、バッファ/マルチプレクサ104と関連する論理はいずれかの標準I/Oインタフェースに準拠していれば良い。たとえば、バッファ/マルチプレクサ104はIDE, ST506, SCSI(すなわち、小型コンピュータシステムインタフェース)及びSA400(すなわち、フロッピーディスク規格)などのI/Oインタフェースの1つ又は2つ以上に準拠していれば良い。従って、その実施例では、パーソナルコンピュータシステム70はそのような標準I/Oインタフ

ェースの中の1つを介して制御装置92と通信すると考えられる。

【0070】図3のパーソナルコンピュータシステム40と、図4のパーソナルコンピュータシステム70は、対応するフラッシュメモリアレイのファイル構造を制御するための専用のハードウェアをそれぞれ含んでいる。パーソナルコンピュータシステム40の場合、その追加ハードウェアはRAMバッファ62である。パーソナルコンピュータシステム70の場合には、その追加ハードウェアは制御装置92である。図3のRAMバッファ62と、図4の制御装置92は、ファイルシステム制御コードをシステムRAM52、フラッシュメモリアレイ64、システムRAM72又はフラッシュメモリアレイ96にそれぞれ記憶する必要をなくするための手段を構成している。

【0071】RAMバッファ62と制御装置92は、図3のシステムRAM52と、図4のシステムRAM72がそれぞれファイル構造58, 94に対するデータバッファとして動作する必要があるように、それぞれデータを緩衝する働きをする。さらに、パーソナルコンピュータシステム40にRAMバッファ62を含め、また、パーソナルコンピュータシステム70に制御装置92を含めることは、システムRAM52及びシステムRAM72がそれぞれフラッシュメモリアレイ64及び96のファイル構造に関するスクラッチパッド領域として動作する必要があるということを意味している。

【0072】図3のRAMバッファ62と、図4の制御装置92とを設けたため、ファイル構造58とファイル構造96を、それぞれ、対応するパーソナルコンピュータシステムに関する基本システムハードウェアの一部であるブート可能記憶領域を含むものとして認識することができる。その結果、RAMバッファ62と制御装置92はそれぞれ対応するパーソナルコンピュータシステムのフラッシュメモリアレイのファイル構造の総体的な性能を向上させる働きをする。

【0073】図2～図4に示すパーソナルコンピュータシステム10, 40及び70は、構成の上で異なっているので、コンピュータの特性も異なる結果になる。図2のコンピュータシステム10では、ファイルシステムドライバソフトウェア28は中央処理装置12により、フラッシュメモリアレイ34を管理するファイル管理ユーティリティの全てを処理するフォアグラウンドタスクソフトウェアルーチンとして実行される。ファイルシステムドライバソフトウェア28はシステムRAM22の一部を占めるが、システムRAM22の大きさは限定されても良いであろう。たとえば、パーソナルコンピュータシステム10のオペレーティングシステム26がMS-DOSである場合、システムRAM22を1メガバイトの「実モード」スペースに限定しても良い。

【0074】図3のパーソナルコンピュータシステム4

0の場合、フラッシュメモリアレイ64のファイル構造を規定するはファイルシステムドライバソフトウェアは、ROM42のRAM BIOSソフトウェア43の一部と、ROM BIOS拡張ソフトウェア44を形成する。しかしながら、ROM42の記憶スペースの量を限定しても良い。たとえば、MS-DOSと互換性をもつパーソナルコンピュータによっては、全ての入出力機能をROMの64キロバイトBIOS部分に含まれているファームウェアによって駆動するものがあり、その場合のROM BIOS拡張ソフトウェアは2キロバイトずつと少なくなっている。ROM42のサイズが小さいならば、ファイルシステムドライバソフトウェアはROM42で利用可能なスペースを越えても良いであろう。

【0075】これに対し、コンピュータシステム70のハードウェア制御装置92はファイルシステムドライバを記憶し、これにより、フラッシュメモリアレイ96、ROM78及びシステムRAM72に関するメモリ要件は緩和される。さらに、ハードウェア制御装置92はファイルシステム制御指令を簡略化する。たとえば、ハードウェア制御装置92は、ファイルシステム制御指令を、他のいずれかのメモリスペースにあるコードを参照するラベルによりソフトウェアアクセス可能である高レベル手順呼出しに変換する。本発明の一実施例においては、簡略化されたファイルシステム制御指令又はファイルドライバ制御指令は従来のBIOS型制御コードとしてハードウェア制御装置92のROM89に記憶される。これにより、ソリッドステートファイルシステム94を基本ブート可能装置として認識することができる。その結果、ファイルシステムの呼出しは簡単になる。ファイルシステムを呼出すたびに、ファイルシステムドライバ（ファイルシステム制御コードともいう）が実行されることになる。

【0076】一実施例では、図4のパーソナルコンピュータシステム70の場合の各ファイルシステム呼出しは、ROM78に記憶されている主BIOS及び直接マッピング方式BIOS拡張ソフトウェアからCPU12が命令を取出すことによって、直接実行される。別の実施例では、各ファイルシステム呼出しは、ROM98に記憶されているBIOS拡張メモリブロックにあるメモリページからCPU12により実行される。さらに別の実施例においては、各ファイルシステム呼出しはハードウェア制御装置92のROM98に記憶されているファイルシステムドライバからマイクロコントローラ100により実行される。

【0077】好ましい実施例では、ハードウェア制御装置92は各ファイル関連タスクをホストCPU12からの単一の指令に短縮し、これがCPU12のオフローディングを増加させるのを助ける。制御装置92は入力データを局所RAMバッファスペース102に保持することにより、入力データをRAM型メモリ速度を受入れ

る。ハードウェア制御装置92は与えられたタスクをそれぞれ自動的に完了するが、一度に1つの動作しか処理しない。

【0078】さらに、制御装置92はホストCPU12に即時データアクセスのためにいずれかのファイルシステムタスクへの割込みを実行させる。これは、そうでなければアクセス不能ブロックにあると思われる有効ファイル情報の追加RAM緩衝を要求する。アクセス不能ブロックは、書き込み動作又は消去動作に関連するブロックである。制御装置92は、割込みに引続いて元のタスクを再開させる。本発明の一実施例においては、ハードウェア制御装置92は所定の数の同時書き込みタスクをキューアップする。

【0079】さらに、一実施例では、制御装置92は、利用可能なフリースペースとホット/コールドブロックサイクルの不均衡を求めて、主フラッシュメモリ記憶装置アレイ96を絶えず監視している。ハードウェア制御装置92は、さらに、必要に応じてフラッシュメモリアレイ96に関する再配分動作を自動的に開始させる。再配分動作はフラッシュメモリアレイ96のブロックの間に循環を均等化させる。再配分動作は必要に応じてハードウェア制御装置92により完全バックグラウンドタスク方式で自動的に開始される。

【0080】ハードウェア制御装置92は、フラッシュメモリアレイ96における記憶のためのスペースを解放するために、フラッシュメモリアレイ96に関するクリーンアップ動作をも自動的に開始する。クリーンアップ動作はハードウェア制御装置92により必要に応じて開始され、完全バックグラウンドタスク方式で実行される。クリーンアップ動作については以下にさらに詳細に説明する。

【0081】ファイルシステム制御のためのハードウェア制御装置92が利用可能なフリースペースと、ホット/コールドブロックサイクルの不均衡とを求めてフラッシュメモリアレイ96を監視し続けると共に、制御装置92がクリーンアップ動作又は再配分動作を自動的に開始させるには、フラッシュメモリアレイ（96）は

(1) パーソナルコンピュータシステム70の内部の記憶装置として常駐しているか、あるいは、(2) フラッシュメモリアレイ96が取外し自在であればパーソナルコンピュータシステム70に対して確実に物理的にインタロックしているべきである。フラッシュメモリアレイ96が取外し自在であるこの別の実施例では、パーソナルコンピュータシステム70は、フラッシュメモリアレイ96に関わるバックグラウンドタスクが発生しているときに点灯するインジケータとしての発光ダイオード（「LED」）を含む。パーソナルコンピュータシステム70のユーザーは、LEDが点灯しているとき、従って、バックグラウンドタスクが発生しているときにはフラッシュメモリアレイ96を取外してはならないとい

う命令を受ける。

【0082】好ましい実施例では、主フラッシュメモリアレイに対するインタフェース回路を(1) 入出力(「I/O」)マッピング、(2) ページング形メモリマッピング又は(3) 直接メモリマッピングすることができる。図5は、インタフェース回路に関するI/Oマッピング方式を示す。

【0083】図2～図4に示す実施例の場合、固定ディスクドライブ又はフロッピーディスクドライブに代わるものとしてフラッシュメモリアレイ34、64及び96を使用できるであろうということを理解すべきである。

【0084】図5に示すI/Oマッピング方式は、フラッシュメモリアレイ81及び局所バッファRAM83に対してマッピングされる直列転送I/Oポートから成るI/Oプレーン85を使用する。図5に示すI/Oマッピング方式は主メモリプレーン(システムRAMともいう)の代わりに別個にI/Oメモリプレーン85を使用するので、図5に示すI/Oマッピング方式はホストコンピュータの主メモリスペースを全く消費しない。すなわち、I/Oマッピング方式はホストコンピュータのシステムRAMを全く消費しないのである。

【0085】ところが、直列転送I/Oポート85に結合するI/O装置は、それらのI/O装置(図示せず)がランダムアクセス装置ではなく、直列装置であると、中央処理装置による直接実行を支援することができない。それらのI/O装置は実行に備えてコードファイルをシステムRAMにダウンロードしなければならない。

【0086】図6は、ページング形マップフラッシュメモリアレイ105に対するページング形メモリマッピングインタフェースを示す。図6は、主メモリプレーン89を示す。主メモリプレーン89は主システムRAM99と、ページングウィンドウ97と、BIOS/ROM拡張ソフトウェア95と、BIOS93と、拡張システムRAM91とを含む。

【0087】本発明の一実施例においては、主メモリプレーン89は1メガバイトのアドレス範囲制約101を有する。主メモリプレーン89について1メガバイトのアドレス範囲制約があるということは、1メガバイトのアドレス範囲制約101を越える主メモリプレーン89の部分をアドレッシングするためにCPU又はマイクロプロセッサの保護モードを使用することを意味する。言い換えれば、CPUの保護モードを使用して、主メモリプレーン89の拡張システムRAM領域91をアドレッシングする。

【0088】初期の世代の典型的な低性能マイクロプロセッサは保護モードを含んでいなかった。さらに、パーソナルコンピュータシステムの初期の世代の典型的な低性能オペレーティングシステムは、保護モードアドレッシングを処理する能力を有していなかった。従って、そのような性能の低いマイクロプロセッサやオペレーティ

ングシステムは1メガバイトのアドレスを越える領域をアドレッシングできなかった。それに反して、後の世代の高性能のマイクロプロセッサやオペレーティングシステムの中には、1メガバイトアドレスを越える領域でもアドレッシングできるものがある。

【0089】図6では、フラッシュメモリアレイ105は小形ページ107、109及び111から構成されており、局所バッファRAM113に結合している。本発明の別の実施例においては、フラッシュメモリアレイ105をどのような数のページからも構成できると考えられる。

【0090】図6のページングウィンドウ97は、ホストの中央処理装置からフラッシュメモリアレイ105のページ107、109及び111への直接アクセスを可能にするページング形メモリマッピングインタフェースである。各ページは必要に応じて異なるフラッシュメモリアレイ105のセグメントを指示するように変更される。

【0091】図6は、装置ドライバを拡張するものとして使用されるページング形ファームウェア103をさらに示す。ページング形ファームウェア103はページ115、117、119及び121を含む。本発明の一実施例では、ページングウィンドウインタフェース97を使用して、ページング形ファームウェア103を指示する。ページングウィンドウ97によってページング形ファームウェア103を指示すれば、さらに大きなファームウェア記憶スペース103への拡張が可能になる。これにより、単一の(通常は2キロバイトの)BIOS/ROM拡張記憶場所95を越える追加の記憶スペースがコンピュータシステムに与えられることになる。

【0092】図7は、主フラッシュメモリアレイへの直接メモリマッピング形インタフェースを示す。図7では、主メモリプレーン135は、主システムRAM137と、BIOS/ROM拡張ソフトウェア139と、BIOS141と、拡張システムRAM145とに加えて、フラッシュメモリアレイマップ149と、局所バッファRAM147とを含む。

【0093】図7に示す主メモリプレーン135は1メガバイトのアドレス範囲制約143を有する。フラッシュメモリアレイマップ149と、局所バッファRAMマップ147はメモリプレーン135の1メガバイトのアドレス範囲制約143を越えたところにある。すなわち、フラッシュメモリアレイマップ149と局所バッファRAMマップ147をアドレッシングするときには、CPU又はマイクロプロセッサの保護モードを使用することになる。これに対し、CPUは主メモリプレーン135の主システムRAM137と、BIOS/ROM拡張ソフトウェア139と、BIOS141については保護モードを使用せずにアドレッシングできる。

【0094】フラッシュメモリアレイマップ149はパ

ーソナルコンピュータシステムのフラッシュメモリアレイに対し直接マッピングする。従って、フラッシュメモリアレイマップ149の大きさはパーソナルコンピュータシステムのフラッシュメモリ記憶アレイと等しい。

【0095】さらに、局所バッファRAMマップ147はパーソナルコンピュータシステムの局所バッファRAMに対し直接マッピングされている。そのため、局所バッファRAMマップ147の大きさはパーソナルコンピュータシステムの局所バッファRAMと等しい。

【0096】このように、直接メモリマッピング方式は、大量の主メモリスペースを使用する。さらに、直接メモリマッピング方式では、1メガバイトのアドレス範囲制約143を越える主メモリブレーンの部分をアドレスリングすることができるCPUとオペレーティングシステムを必要とする。

【0097】直接マッピング方式によれば、フラッシュメモリアレイの全てから直接コード実行が可能である。これに対し、ページング形マッピング方式ではフラッシュメモリアレイの一部からの直接コード実行が可能である。

【0098】直接マッピング方式は1メガバイトの制約を越えてアドレスリングすることができる。CPUとオペレーティングシステムを要求する。これに対し、ページング形マッピング方式は、1メガバイトの制約を越えてアドレスリングすることが不可能であるCPU及びオペレーティングシステムと共に動作する。

【0099】図8、図11、図12及び図14から図16は、パーソナルコンピュータシステムのフラッシュメモリEPROMアレイについて好ましいファイル構造の例を示している。特に、図8、図11、図12及び図14から図16は、図2から図4のパーソナルコンピュータシステム10、40及び70のフラッシュメモリアレイ34、64及び96について好ましいファイル構造をそれぞれ示している。図8、図11及び図12に示すファイル構造は可変ファイル構造であり、図14から図16に示すファイル構造はセクター方式ファイル構造である。図8及び図14に示すファイル構造は、それぞれ、動的ファイルディレクトリを含み、これはフラッシュメモリアレイの中にある。図11、図12、図15及び図16に示すファイル構造は、それぞれ、別個のEEPROM又はスタティックRAM（「SRAM」）に記憶されるファイルディレクトリを含む。図8、図11、図14及び図15に示すファイル構造は連係リストディレクトリエントリを含む。図12及び図16に示すファイル構造は、それぞれ、従来のハードディスク又は固定ディスクに使用されていたディレクトリに類似するファイルディレクトリを含む。可変ファイル、セクター方式ファイル、動的ディレクトリ、別個に記憶されるディレクトリ、連係リストディレクトリ及びディスク類似のディレクトリについては、以下にさらに詳細に説明する。図

8、図11、図12及び図14から図16に示すファイル構造は、それぞれ、クリーンアップ動作の直後の状態にある。

【0100】図8は、フラッシュメモリアレイ112のファイル構造110を示す。ファイル構造110は可変サイズファイル構造である。可変サイズファイル構造を可変ファイル構造ともいう。

【0101】ファイル構造110のような可変ファイル構造においては、パーソナルコンピュータのCPUのプログラムカウンタがコード又はデータを通してステップできるように、コード又はデータを間断なく記憶してゆく。従って、可変ファイル構造は現場実行を可能にする。現場実行とは、CPUがRAMを必要とせずにフラッシュメモリアレイから直接にコードを実行できることを意味する。現場実行を直接コード実行又は現場コード実行ともいう。

【0102】ファイル構造110は可変ファイル構造であるため、直接現場実行動作に備えてコードファイルを大きさに関係なく間断なく記憶することができる。現場実行動作によって、実行に備えてRAM型メモリにダウンロードするための時間を費やさなくて済むようになる。従って、フラッシュメモリに可変ファイル構造を使用すると、パーソナルコンピュータシステムにおけるRAM要件は緩和される。その結果、このようなファイル構造を使用しない場合に必要になると思われる余分なRAMに費やされる分だけ、システムのパワーを節約できる。

【0103】ファイル構造110を適正に定様式化した場合、主アレイにあるヘッダを除いて、ホットブロック又はコールドブロックは存在していない。すなわち、電氣的データ妨害状態を防止するための隣接ブロックの消去／再プログラミング循環を追跡する必要はないのである。

【0104】一方、フラッシュメモリアレイの可変ファイル構造は従来の典型的なディスクドライブに見られるセクターに基づくファイル構造とはかなり異なっている。パーソナルコンピュータの従来の典型的なオペレーティングシステムは、ディスクセクター方式ファイル構造と組合わせた場合にのみ動作するように設計されている。従って、可変ファイル構造と共に動作するためには、現在の典型的な高レベルオペレーティングシステムを修正するか又は追加しなければならないと考えられる。

【0105】さらに、可変ファイル構造の場合、記憶スペースの再割付けのためのファイルクリーンアップ動作（以下にさらに詳細に説明する）はセクター方式ファイル構造より効率が悪く且つより頻繁である。従って、可変ファイル構造では、クリーンアップ動作はより長いバックグラウンド時間を要する。さらに、可変ファイル構造はより多くの総合フラッシュメモリ消去／書き込み循環

を必要とする。

【0106】フラッシュメモリアレイ112は、コード又はデータを記憶する多数のデータブロックを含む。図8は、クリーンアップ動作直後の状態にあるフラッシュメモリアレイ112のファイル構造の1例を示す。

【0107】フラッシュメモリアレイ112のブロック114、116及び118は、予約ブロック又は予約済みブロックとも呼ばれる「予約となるべき」ブロックである。予約ブロックは、クリーンアップ動作中に一時ファイルバックアップを実行するブロックである。クリーンアップ動作は再割付けのために使用される。クリーンアップ動作は、まず、最終的に消去されるブロックから現在活動中である全てのファイルを取外すことから成る。各フラッシュメモリブロックは全て一度に消去できる。従って、予約ブロック114、116及び118はクリーンアップ動作中の消去／再割当てのために利用可能なブロックである。このように、予約ブロック114、116及び118は一時ファイルバックアップを実行する。

【0108】一実施例では、予約ブロックの中の少なくとも1つを予備予約ブロックとして使用する。この予備ブロックは、クリーンアップ動作のために確実に1つの予約ブロックを利用できることを保証するのを助け、それにより、ファイルシステムの信頼性を向上させるのを助ける。

【0109】図8に示すように、ブロック120は空きとなるべきブロックである。

【0110】ブロック122、124、126及び128は空きブロックである。空きデータブロック122、124、126及び128は、コード及びデータを記憶するために使用可能なブロックである。図8の空きブロック122、124、126及び128はアクティブブロックではなく、コード又はデータを記憶するために使用できるブロックである。

【0111】パーソナルコンピュータシステムの動作中、空きブロックにデータ又はコードが導入される。データブロック130は部分空きブロックである。ブロック130の一部はアクティブコード又はアクティブデータを含む。ブロック130の残る部分は一部が空いており、後にコード又はデータを記憶するためにその空き部分を利用できる。

【0112】空きブロックや部分空きブロックがコードとデータで充填されてゆくにつれて、データ又はコードを含まないままであるブロックの数は減少する。使用可能な空きブロックの数が予約ブロックの許容数に達すると、クリーンアップ動作が始まる。

【0113】ブロック132、136、138、140、142及び144は、ブロック130の一部と共に、アクティブ連続可変サイズファイル及び削除連続可変サイズファイルを構成する。一実施例では、ブロック

132、136、138、140、142及び144はブロック130の一部と共にアクティブサブディレクトリ及び削除サブディレクトリをさらに含む。

【0114】ファイル150、164、166、168、170、172及び174はブロック132、136、138、140、142及び144の中の削除ファイルである。図8に示す削除ファイル150、164、166、168、170、172及び174は、ユーザーが削除してしまったファイルである。

【0115】クリーンアップ動作中、アクティブファイルの中で最も古いものをブロック114及び116に再び書込む。図8に示すファイル構造110の場合、ブロック138、140、142及び144は最も古いアクティブ連続可変ファイル及び削除連続可変ファイルを含む。ブロック138、140、142及び144の中に見られるアクティブファイルは、ブロック114及び116に再び書込むべきファイルである。クリーンアップ動作中には、4つのブロック138、140、142及び144を2つの予約ブロック114及び116にはめ込むことができるように、圧縮動作が起こる。圧縮動作中、先にユーザーが削除したファイルはブロック114及び116に書込まれない。図8では、ブロック138、140、142及び144の中に見られる削除ファイルはファイル168、170、172及び174である。従って、それらの削除ファイル168、170、172及び174はブロック114及び116には再び書込まれない。ブロック138、140、142及び144は次に圧縮すべき領域にあるという。圧縮可能なスペースの量は、削除ファイル168、170、172及び174が占めるスペースの量である。

【0116】ブロック114、116、118及び120は、先にクリーンアップ動作の結果として発生したブロックである。従って、ブロック114、116、118及び120は、最前のクリーンアップ動作の結果として発生した圧縮スペースを構成しているという。

【0117】図8に示すファイル構造のような可変ファイル構造についてのクリーンアッププロセスは循環方式である。予約ブロックは、フラッシュメモリアレイ112の物理的最上部からその物理的最下部へループしてゆく場合を除いて、連続する順序である。先に述べた通り、使用可能な空きブロックの数が予約ブロックの許容数に到達すると、クリーンアップ動作が始まる。アクティブファイル及び削除ファイルを含む最も古いブロックを、まず最初に圧縮し、クリーンアップする。連続するアクティブ可変ファイルと削除可能ファイルを含む最も古いブロックは、フラッシュメモリアレイ112の中の予約ブロックのすぐ後に見出される。クリーンアップ動作の一部として再び書込むべきブロック及び圧縮すべきブロックの数は、それらのブロックの中の削除ファイルの数によって決まる。言い換えれば、最も古いブロック

の中の圧縮可能スペースが、1回のクリーンアップ動作の一部として再び書込まれるそれらのブロックの中のブロック数を決定する。たとえば、ブロック138, 140, 142及び144は4つのブロック138, 140, 142及び144を2つのブロック114及び116に圧縮させるのに十分な数の削除ファイルを含む。

【0118】ブロック138, 140, 142及び144をブロック114及び116に圧縮した後、ブロック138, 140, 142及び144を消去する。ブロック138, 140, 142及び144の消去後、ブロック138, 140及び142は予約ブロックになり、ブロック144は空きブロックになる。

【0119】図8に示すファイル構造110は、フラッシュメモリアレイ112の内部のファイルの属性と場所を記憶するディレクトリを含む。ファイル構造110の場合、全てのディレクトリ情報はフラッシュメモリアレイ112のブロック134の中に記憶される。そこで、ファイル構造110のディレクトリをフラッシュメモリアレイ112のディレクトリという。

【0120】ファイル構造110のディレクトリは動的ディレクトリ再配置を要求する。動的ディレクトリ再配置とは、ファイルシステムが書き込み動作と消去動作によってディレクトリをフラッシュメモリアレイ112の中に様々な場所へ移動させることを意味する。図8に示す例の場合、ディレクトリはフラッシュメモリアレイ112のブロック134を占める。ところが、様々な時点で、ディレクトリは、ブロック114, 116, 118, 120, 122などを含めたフラッシュメモリアレイ112のその他のブロックの中のいずれかを占めることができるであろう。

【0121】図8に示す特定の時点においては、ディレクトリはブロック134にある。ブロック134はディレクトリフラグ152と、ブート記録154と、ブロック状態テーブル156（BST156ともいう）と、連係リストディレクトリエントリ158と、空きディレクトリスペース160とを含む。このブロックをディレクトリブロックともいう。

【0122】ディレクトリフラグ152は、どのブロックがファイル構造110に関するルートディレクトリを記憶するかを指示するデータのパターンを記憶する。本発明の一実施例においては、ディレクトリフラグ152は、ディレクトリフラグ152を含んでいるブロックがルートディレクトリを含むブロックでもあることを指示する独自のランダムでないデータパターンを記憶する。たとえば、一実施例では、ディレクトリフラグ152は一連の反復AAh/55hデータを記憶する。別の実施例では、ディレクトリフラグ152は他の何らかのランダムでないデータのシーケンスを記憶する。

【0123】本発明の一実施例においては、ある特定のブロックが実際にディレクトリブロックであるのか否か

を判定するために補足検査を実行する。これは、ディレクトリブロック以外のブロックがディレクトリフラグ152のデータパターンと一致する開始データパターンを有していた場合に起こると思われる、ディレクトリブロックの場所を規定する際に誤りを回避するのを助けるために実行される。補足検査は、ディレクトリブロック134のブートレコード154の中にある検査合計データフィールドを検査することにより実行される。一実施例では、ブートレコード154の検査合計データフィールドは、ブートレコード154が有効であり且つ現在記録である場合には変化しないことから選択された特定のデータのパターンである。別の実施例においては、ブートレコード154の検査合計データフィールドは専用サイクル冗長性検査（「CRC」）又は誤り修正コード（「ECC」）データフィールドである。まず、コンピュータシステムはディレクトリフラグを探索し、次に、検査合計データフィールドを検査することになるであろう。ブートレコード154の検査合計データフィールドの検査から正しい結果が得られると、コンピュータシステムはフラッシュメモリアレイ112に関する現在有効ディレクトリブロックを実際に見出したことを知るであろう。

【0124】本発明の一実施例においては、ブートレコード154が有効でなくなるか又は現在レコードでなくなるときには、ブートレコード154の検査合計データフィールドの論理値1を論理値0に重ね書きするであろう。

【0125】ブートレコード154はブロック134の内部の、ディレクトリフラグ152の隣に位置している。ブートレコード154は特定のファイルシステムの型及び特定のファイルシステムの改訂に関する情報をレコードする。ブートレコード156はいくつかのファイルシステムパラメータをさらに含んでいる。それらのパラメータには、定様式化可能容量情報、予約ブロック情報、付属ディレクトリ情報、旧ディレクトリに関連する情報及びルートディレクトリに関連する情報がある。

【0126】定様式化可能容量情報はフラッシュメモリアレイ112のコード及びデータを記憶するために定様式化することができる領域の大きさである。予約ブロック情報は、どのブロックが予約ブロックであるかを識別する。先に述べた通り、予約ブロックはクリーンアップ動作及び再割付け動作の間に使用されるブロックである。付属ディレクトリ情報は、追加ディレクトリ情報の記憶のために使用されている追加ブロックの場所である。1つ又は複数の追加ブロックに記憶されている付属ディレクトリは、フラッシュメモリアレイ112の内部に記憶されているファイルの数がある量を越えた場合に必要になるであろう。旧ディレクトリは現在ディレクトリのバックアップコピーである。旧ディレクトリに関連するパラメータ情報は、フラッシュメモリアレイ112

の中の旧ディレクトリが記憶されている場所である。たとえば、図8では、旧ディレクトリはブロック114に記憶されている。ルートディレクトリは第1のディレクトリエントリである。ルートディレクトリに関連するパラメータ情報はフラッシュメモリアレイ112の中のルートディレクトリの場所である。

【0127】ブロック状態テーブル156は、どのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥ブロック又は故障ブロックであるかに関する情報を含む。一実施例では、ブロック状態テーブル156は各ブロックに関するサイクルカウント情報をさらに含む。サイクルカウント情報は、ある1つのブロックが消去され、再書き込みされた回数を指示する。ブロック状態テーブル156はブロック134の中のブートレコード154の隣に位置している。

【0128】連係リストディレクトリエントリ158を主ディレクトリ158ともいう。主ディレクトリ158はブロック134の中のブロック状態テーブル156の隣に位置している。

【0129】主ディレクトリ158は一連の個別ディレクトリエントリを含む。各ディレクトリエントリはディレクトリエントリの型、すなわち、ディレクトリエントリが(1) ファイルに関わるものであるか又は(2) サブディレクトリに対するポインタであるかに関する情報を含む。

【0130】各ディレクトリファイルエントリは、ファイル名/拡張と、日時と、ファイルの型と、ファイルの長さ、フラッシュメモリアレイ112における物理ファイルアドレスと、シブリングポインタと、チャイルドポインタとに関する情報を含む。

【0131】本発明の一実施例においては、ファイル名/拡張は、ユーザーにファイルを識別させる10バイトである。その実施例の場合、ファイル名/拡張は語、数及び文字の形をとることができる。

【0132】ファイルエントリ中の日時情報は、ファイルが作成された日付、もしくはファイルが最後に改訂又はセーブされた日付を指定する。

【0133】ファイル型情報は、ファイルがコードから成るか、ページ形コードから成るか、データから成るか又はデータ付属セグメントから成るかを表わす。

【0134】ファイル長さ情報はファイル又はセグメントの物理的大きさを表わす。物理的ファイルアドレス情報はフラッシュメモリアレイ112の中におけるファイル又はセグメントの場所を表わす。

【0135】シブリングポインタとチャイルドポインタは連係リストディレクトリ階層から構成される。言い換えれば、シブリングポインタとチャイルドポインタはディレクトリ構造ツリーから構成されるのである。

【0136】各ポインタは別のディレクトリエントリの

場所を指示する。本発明の一実施例においては、ポインタはフラッシュメモリアレイ112の中の指示されるディレクトリエントリの絶対物理アドレスである。ところが、本発明の別の実施例では、各ポインタは、そのディレクトリを記憶しているブロックにおける増分アドレスを表わすオフセットアドレスから構成されている。

【0137】絶対物理アドレスから成るポインタの1例が16進数の1264FFFFである。これに対し、オフセットアドレスから成るポインタの1例は16進数のFOFである。ポインタのオフセットアドレスをインデックスアドレスともいう。

【0138】ポインタについてインデックス方式を採用する実施例では、ポインタは最小のサイズに抑えられる。また、インデックス方式によれば、クリーンアップ動作後にディレクトリを再配置してから、各ポインタアドレスを計算し直す必要はない。

【0139】フラッシュメモリアレイ112のブロック1つではディレクトリ全体を記憶しきれない場合には、ルートディレクトリにより単一の二次付属ディレクトリブロック又は2つ以上の二次付属ディレクトリブロックを指示することができる。1つ又は複数の付属ディレクトリブロックを使用するときには、各「インデックス型」ポインタは、ディレクトリエントリポインタがディレクトリブロックの中のどれを参照しているかを確定するためのインデックスを含んでいるものと考えられる。

【0140】本発明の一実施例では、全てのディレクトリエントリを共通の長さにしなけなければならないようになっており、それにより、ポインタのインデックスアドレスのサイズはさらに縮小される。この共通の長さはインデックスが参照するオフセットアドレス増分に対応する。

【0141】ディレクトリエントリ158は連係リストを形成する。ファイルシステムディレクトリツリーの全体は、ルートディレクトリエントリを開始点として、シブリングポインタ及びチャイルドポインタを使用しながらディレクトリエントリを連係リストと連鎖させてゆくことにより構成される。連係リストディレクトリツリーの例を図9及び図10に示す。図9及び図10に示されているディレクトリエントリは、ルートディレクトリエントリ157と、エントリA181と、エントリB183と、エントリC185と、エントリD189と、エントリE191と、エントリF193と、エントリG197と、エントリH199と、エントリI187と、エントリJ195である。チャイルドポインタはポインタ159、161、165、169及び171である。また、シブリングポインタはポインタ163、167、171、173、175及び179である。

【0142】図9は、連係リストディレクトリエントリの論理的配列を示す。図10は、フラッシュメモリアレイ112のブロック134の中における連係リストディ

レクトリエントリの物理的場所を示す。

【0143】ディレクトリエントリのどの連係リストチェーンにおいても、その最後のディレクトリエントリのシブリングポインタフィールド及びチャイルドポインタフィールドは、全てが論理値1であるF（16進数）ナルデータパターンを含む。これは、そのディレクトリエントリがチェーンの終端にあることを示している。（消去されたディレクトリエントリはその全体、すなわち、その全てのフィールドにF（16進数）ナルデータパターンを含む。）

【0144】ファイルを作成、移動又は変更するたびに、連係リストチェーンにディレクトリエントリを追加する。特定の連係リストチェーンにおける最終ディレクトリエントリのシブリングポインタフィールド又はチャイルドポインタフィールドのF（16進数）ナルデータパターンは、新たに作成されるディレクトリエントリに対するシブリングポインタ又はチャイルドポインタである新たなデータパターンに重ね書きされる。フラッシュメモリアレイの個々のビットセルを論理値1から論理値0に重ね書きすることは可能であるので、この重ね書きを実行することができる。従って、ファイルが作成、移動又は変更されるにつれて、ディレクトリエントリ158の連係リストチェーンは成長し、空きスペース160の中へ移動する。言い換えれば、空きディレクトリスペース160は時間の経過に伴う連係リストディレクトリエントリ158の拡張のための余地を与える。

【0145】各ディレクトリエントリは、そのファイル又はサブディレクトリエントリの有効性を指示するための状態ビットをさらに有する。それらの状態ビットは、ディレクトリエントリが単にディレクトリの連係を継続してゆくだけであるが、アクティブファイル又はサブディレクトリを表わさなくなることを指示するために、0の状態に重ね書きされる。単なる連係を指示するために状態ビットが0の状態に重ね書きされた時点で、ディレクトリエントリはフラッシュメモリアレイ112内部の未使用スペースであると考えられる。さらに、ディレクトリエントリがファイル型エン트리であり且つエン트리が単にディレクトリの連係を継続してゆくだけであること指示するために状態ビットが0の状態に重ね書きされた場合には、このファイル型ディレクトリエントリと関連するフラッシュメモリアレイ112の中のスペースは未使用スペースであると考えられる。

【0146】その後の書き込みのためにこの未使用スペースを解放するため、ファイルシステムのクリーンアップが必要である。フラッシュメモリブロックは全て一度に消去されるので、それぞれの未使用ディレクトリエントリを個別に再書き込みすることは不可能である。動的ディレクトリ再配置はこの問題を克服するのを助ける。

【0147】クリーンアップ中、未使用のディレクトリエントリを圧縮する。圧縮される未使用のディレクトリ

エントリを伴う新たなディレクトリ連係リストは、全てルートディレクトリエントリから有効に始まらなければならない。動的ディレクトリ再配置は、別個のブロックにおける新たな連係リストを再構成することを伴う。先のディレクトリブロックは、新たなディレクトリツリーが構成されるまでバックアップとして働く。新たなディレクトリツリーは、新たな付属ディレクトリブロックと共に新たなディレクトリから構成されても良いであろう。本発明の一実施例においては、ディレクトリ再配置の一部であるディレクトリ構成の間に、新たなディレクトリをフラッシュメモリアレイ112の予約ブロックに直接に書き込む。たとえば、フラッシュメモリアレイ112の予約ブロック116に新たなディレクトリを書き込むことができるであろう。

【0148】クリーンアップに際しては、まず、最終的に消去されるブロックから現在活動中である全てのファイルを取り出すが、それが再配置である。新たなディレクトリは旧ファイルの圧縮（すなわち、再移動）と、フラッシュメモリアレイ112における依然として活動中のファイルの物理的再配置の双方を反映している。本発明の一実施例では、それらの転送ファイルは旧フラッシュメモリ記憶場所から新たなフラッシュメモリ記憶場所に直接に書き込まれる。

【0149】ブロック状態テーブル156は、どのブロックが空き、使用中、予約中又は欠陥ブロックであるかを追跡するために使用される。ブロック状態テーブル156は通常のファイル読取り動作中、通常のファイル書き込み動作中及びクリーンアップ動作中にブロックを追跡する。

【0150】本発明の別の実施例においては、RAM緩衝を利用する。RAM緩衝を利用した場合、クリーンアップ動作中、ファイルがフラッシュメモリアレイ112の中の新たな記憶場所に書き込まれる前に、即時ステップとして、パーソナルコンピュータシステムのRAMにファイルを書き込む。RAM緩衝の場合には、転送ファイルはフラッシュメモリアレイ112の旧記憶場所から新たな記憶場所へ直接には書き込まれない。その代わりに、ファイルはパーソナルコンピュータシステムのRAMを通過する。

【0151】本発明の別の実施例では、いくつかのディレクトリエントリが付属ポインタをさらに含んでいる。付属ポインタは個々のデータファイルを2つのデータファイルに分割するために使用される。その一方のデータファイルがある1つのブロックにあるとき、他方のデータファイルは別のブロックにある。これによりデータファイルをブロック境界のあたりで分割することができる。また、個々のデータファイルはフラッシュメモリアレイ112の物理アドレスの終わりから始めへループすることができる。さらに、個々のデータファイルは欠陥のある、すなわち、不良のブロックのあたりで飛越すこ

とができる。

【0152】本発明のさらに別の実施例においては、ファイルの各ディレクトリエントリはファイル名、接尾部及びアドレスに関する二次フィールドをさらに含む。これにより、元のディレクトリエントリを放棄する必要なく、ファイルの再名前付け、編集及び追加が一度に可能になる。たとえば、データファイルを編集するときにバックアップコピーを保持する場合に接尾部再名前付けを使用できるであろう。

【0153】本発明の別の実施例では、図8に示すファイル構造110はヘッダをさらに含む。ヘッダは、たとえば、ブロック122にあり、その場合、ブロック122は空きブロックではなくなる。

【0154】ヘッダは、ディレクトリ及びフラッシュメモリアレイ112に関する情報を含むファイルである。一実施例では、ヘッダはフラッシュメモリアレイ112全体の未定様式化サイズと、フラッシュメモリアレイ112の定様式化済サイズと、フラッシュメモリアレイ112中のブロックの総数とに関する情報並びにフラッシュメモリアレイ112に関する詳細な装置情報とを含む。本発明の一実施例においては、ヘッダに記憶される詳細な装置情報は消去電圧及び書込み電圧と；消去指令及び書込み指令と；消去アルゴリズム及び書込みアルゴリズムと；個別最大循環仕様及び全チップ最大循環仕様と；フラッシュメモリアレイ112の読取り性能特性、書込み性能特性及び消去性能特性とを含む。

【0155】別の実施例では、ヘッダは1つ又は複数の代替ヘッダの場所に関する情報をさらに含む。

【0156】ヘッダがフラッシュメモリアレイ112のブロックの中の1つ、たとえばブロック122に記憶されているこの実施例の場合、ヘッダを周期的に再生しなければならない。ヘッダの周期的再生はヘッダを周期的に完全に消去し且つ再書込みすることを伴う。このようなヘッダの周期的再生が必要であるのは、通常、ヘッダがフラッシュメモリアレイ112のその他のブロックに対してコールドブロックであるためである。言い換えれば、ヘッダはフラッシュメモリアレイ112のその他のブロックと比べて循環頻度の少ないブロックにある。従って、ヘッダを周期的に再生しなければならないのである。

【0157】ヘッダをどの時点で再生しなければならないかを判定するために、ヘッダに隣接するブロックと、ヘッダの消去／プログラムサイクルの数を追跡する。主ヘッダブロックにはサイクルカウントパラメータが記憶されている。サイクルカウントはヘッダと、ヘッダに隣接するブロックの消去／書込みサイクルの数を表わす。

【0158】別の実施例では、ヘッダの代替（すなわち、バックアップ）コピーをフラッシュメモリアレイ112の別のブロック、たとえばブロック124に記憶する。ブロック124に記憶されるこの代替ヘッダは、再

生動作が起こっている間にヘッダとして一時的に使用される。

【0159】さらに別の実施例においては、ヘッダはフラッシュメモリアレイ112の中にあるのではなく、パーソナルコンピュータシステムの一部を成す別個の集積回路カードにあるメモリアレイの中に記憶されている。ヘッダを記憶しているメモリアレイは「レジスタ番号でアクセスされる」メモリアレイであると考えられる。このように、ヘッダは別個の陰のアレイに記憶されるのである。

【0160】この陰のアレイを使用する実施例の場合、ヘッダはフラッシュメモリアレイ112の中にはないので、ヘッダを再生する必要はないであろう。従って、ヘッダはコールドブロックではなく、また、ホット／コールド循環に関連するサイクルカウントパラメータを含んでいる必要はないと考えられる。

【0161】図8に示すファイル構造110を参照すると、ヘッダの使用は任意であることがわかるはずである。本発明の一実施例は次のような構造によってヘッダの使用を回避している。フラッシュメモリアレイ112はメーカー識別コード及び装置識別コードを記憶している。本発明の一実施例では、そのようなメーカー識別コード及び装置識別コードに基づいて、パーソナルコンピュータシステムのソフトウェアは、ヘッダ型情報を記憶しているルックアップテーブルを参照し、それらの特定のコードからヘッダの型を推論することができる。

【0162】図11は、フラッシュメモリアレイ212のファイル構造210を示す。ファイル構造210は可変サイズファイル構造（可変ファイル構造ともいう）である。フラッシュメモリアレイ212のブロック214、216及び218は「予約となるべき」ブロック（予約ブロック又は予約済ブロックともいう）である。ブロック220は空きとなるべきブロックである。

【0163】ブロック222、224、226及び228は空きブロックである。ブロック230は、一部にデータ又はコードを含み、一部が空いている一部空きブロックである。

【0164】ブロック232、233、236、238、240、242及び244は、ブロック230の一部と共に、アクティブ連続可変サイズファイル及び削除連続可変サイズファイルを構成する。一実施例では、ブロック232、233、236、238、240、242及び244は、ブロック230の一部と共に、アクティブサブディレクトリ及び削除サブディレクトリをさらに含む。ファイル250、264、266、268、270、272及び274はブロック232、236、238、240、242及び244の中の削除ファイルである。

【0165】図11に示すファイル構造210の場合、ブートレコード254と、ブロック状態テーブル256

と、連係リストディレクトリエントリ258と、空きディレクトリエントリ260は、フラッシュメモリアレイ212の一部ではない別個のメモリ234に記憶される。ただし、この別個のメモリ234は、フラッシュメモリアレイ212をも含むパーソナルコンピュータシステムの一部として含まれている。

【0166】本発明の一実施例では、別個のメモリ234は従来のバイト可変電氣的消去可能プログラマブル読取り専用メモリ（「EEPROM」）である。

【0167】本発明の別の実施例においては、別個のメモリ234はスタティックRAM（「SRAM」）である。そのSRAM234は、パーソナルコンピュータシステムの一部であるバッテリーに結合している。パーソナルコンピュータシステムへの外部からの電力が供給されなくなったとき、SRAM234に結合しているバッテリーはSRAM234に電力を供給し続ける。バッテリーによりSRAM234に電力が印加されると、SRAM234は、パーソナルコンピュータシステムへの給電が停止している間でもデータやコードを保持できる。バッテリーがないと、SRAMへの給電が止まるとSRAMは全てのコードとデータを失ってしまうので、バッテリーは必要である。

【0168】本発明のさらに別の実施例においては、メモリ234はNVRAMから構成されている。

【0169】図11に示す実施例では、メモリ234はEEPROMである。定義上、連係リストディレクトリエントリ258がメモリ234の中に見出されれば、EEPROM234にディレクトリフラグを記憶する必要はない。言い換えれば、メモリ234には現在ディレクトリが記憶されている。

【0170】本発明の一実施例では、ファイル構造210はフラッシュメモリアレイ212の中に記憶されているヘッダを含む。ヘッダは、先に図8に関して論じたヘッダに含まれている情報と同様な情報を含む。図11のファイル構造210に関しては、ヘッダは、ルートディレクトリがメモリ234の記憶場所258の中の一定の記憶場所にあることを示す情報をさらに含む。言い換えれば、ヘッダはルートディレクトリをメモリ234内部の固定物理アドレススペースにあるものとして規定する。同じように、ディレクトリフラグは不要である。

【0171】先の述べた通り、フラッシュメモリアレイ212はブロック単位でのみ消去可能であり、個々のバイトごとに消去されるのではない。これに対し、EEPROM234はバイトごとに電氣的に消去可能であり且つユーザー側で再書き込み可能である。従って、EEPROM234はバイト可変であるという。EEPROM234がバイト可変であるということは、連係リストディレクトリエントリ258の中のファイル名、接尾部、アドレス及びポイントを個々に消去、再書き込み及び変更できることを意味している。直接個別変更を先行するシブ

リングポイント及びチャイルドポイントとなるようにすることにより、削除ファイルの飛越しが可能である。

【0172】図11に示すファイル構造210の実施例では、連係リストディレクトリエントリ258の中の各ディレクトリエントリは連係リストディレクトリエントリ258の中のその他のディレクトリエントリのそれぞれとほぼ同じ大きさであると規定されている。そこで、削除したディレクトリエントリを必要に応じてランダムに再利用することができる。

【0173】図11の連係リストディレクトリエントリ258は、その他の点では、図8の連係リストディレクトリエントリ158と同様である。連係リストディレクトリエントリ258は、先に連係リストディレクトリエントリ158に関連して説明してインデックス型ポイントを支援する。

【0174】図11のブートレコード254は図8のブートレコード154に類似している。図11のブートレコード254は特定のファイルシステムの型と、特定のファイルシステムの改訂とに関する情報を含む。ブートレコード254は、フラッシュメモリアレイ212の定様式化可能容量と、フラッシュメモリアレイ212に関する予約ブロック情報と、連係リストディレクトリエントリ258内におけるルートディレクトリの場所に関連する情報とを含むいくつかのファイルシステムパラメータをさらに含む。

【0175】図11のブロック状態テーブル256は図8のブロック状態テーブル156に内容の上では類似している。図11のブロック状態テーブル256はフラッシュメモリアレイ212のどのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥又は故障ブロックであるかに関する情報を含む。

【0176】図11のフラッシュメモリアレイ212に関するクリーンアップ動作は、図8のフラッシュメモリアレイ112に関するクリーンアップ動作に類似している。図11のフラッシュメモリアレイ212のクリーンアップはブロック状態テーブル256と関連して進行し、先に図8のフラッシュメモリアレイ112に関して説明したように循環方式で実行される。図11に示すファイル構造210の場合、ブロック238、240、242及び244は次にクリーンアップし且つ圧縮すべきブロックである。

【0177】図11の空きディレクトリスペース260は、将来、連係リストディレクトリエントリを拡張させてゆくことができるスペースである。ディレクトリエントリ258は、（１）メモリ234がバイト可変EEPROMである場合はバイト可変であり、（２）メモリ234がRAMである場合にはビット可変であるということを理解すべきである。

【0178】図12は、フラッシュメモリアレイ302

のファイル構造300を示す。ファイル構造300は、ハードディスクのディレクトリエントリやフロッピーディスクのディレクトリエントリに類似するバイト可変ディレクトリエントリ366を含む。バイト可変ディレクトリエントリ366は別個のメモリ360に記憶されている。

【0179】ファイル構造300は、可変ファイル構造とも呼ばれる可変サイズファイル構造である。

【0180】フラッシュメモリ302のブロック322、324及び326は「予約となるべき」ブロック（予約済ブロック又は予約ブロックともいう）である。ブロック328は空きとなるべきブロックである。

【0181】ブロック330、304、306及び308は空きブロックである。ブロック310は、一部にデータ又はコードを含んでいる一部空きブロックである。

【0182】ブロック312、314、316、318及び320は、ブロック310の一部と共に、（1）アクティブ連続可変サイズファイル及び削除連続可変サイズファイルと、（2）固定長アクティブサブディレクトリファイル及び固定長削除サブディレクトリファイルとを構成する。ファイル340、342、344、346、350及び354はブロック312、314、316、318及び320の中の削除ファイルである。ファイル338、348及び352はブロック316、318及び320の中の削除サブディレクトリファイルである。

【0183】図12に示すファイル構造310の場合、ブート記録362と、ブロック状態テーブル364と、ディレクトリエントリ366は、フラッシュメモリアレイ302の一部ではない別個のメモリ360に記憶される。ただし、その別個のメモリ360は、フラッシュメモリアレイ302をも含むパーソナルコンピュータシステムの一部として含まれている。

【0184】本発明の一実施例では、別個のメモリ360は従来のバイト可変EEPROMである。別の実施例においては、別個のメモリ360はSRAMであり、そのSRAM360はパーソナルコンピュータシステムの一部であるバッテリーに結合している。パーソナルコンピュータへの外部からの給電が停止すると、SRAM360に結合しているバッテリーはSRAM360に電力を供給し続ける。

【0185】本発明のさらに別の実施例では、メモリ360はNVRAM（すなわち、不揮発性RAM）から構成されている。

【0186】図12に示す実施例では、メモリ360はバイト可変EEPROMである。定義上、ディレクトリエントリ366がメモリ360の中に見出されるのであれば、EEPROM360にディレクトリフラグを記憶する必要はない。現在ルートディレクトリはメモリ360に記憶される。

【0187】一実施例においては、ファイル構造300はフラッシュメモリアレイ306に記憶されているヘッダを含む。このヘッダは図8に関連して論じたヘッダに類似している。図12のファイル構造300の場合のヘッダは、メモリ360の記憶場所366の中の一定の記憶場所にルートディレクトリがあることを示す情報をさらに含む。同様に、ディレクトリフラグは不要である。

【0188】フラッシュメモリアレイ302はブロック単位でのみ消去可能であり、個々のバイトごとに消去することはできない。これに対し、EEPROMは電氣的に消去可能であり、バイトごとにユーザー側で再書き込み可能である。従って、EEPROM360はバイト可変であるという。EEPROM360がバイト可変であるということは、ディレクトリエントリ366を個別に消去、再書き込み及び変更できることを意味する。

【0189】ファイル構造300のディレクトリ構造366は、パーソナルコンピュータシステムにおけるハードディスク又はディスクットのディレクトリ構造に類似している。ディレクトリ構造366をディスク類似ディレクトリ構造と呼ぶ。

【0190】ディスク類似ディレクトリ構造366によって、パーソナルコンピュータシステムについて、磁気ハードディスクに基づくファイル構造との間に偽似互換性をもつオペレーティングシステムインタフェースが得られる。ファイル構造300の場合、ディレクトリとサブディレクトリの双方の内容を記憶するために、主フラッシュメモリアレイ302でデータファイルを作成する。同様に、パーソナルコンピュータシステムの従来の磁気ハードディスクドライブとフロッピーディスクドライブにおいては、ディレクトリとサブディレクトリの双方の内容をディスクに記憶することになる。さらに、ディレクトリエントリ366は、従来の数多くのハードディスクドライブに記憶されているDOSオペレーティングシステムの中に見出されるファイル割付けテーブル（「FAT」）に類似する構造を含む。

【0191】図12に示す実施例の場合、ディレクトリエントリ366はメモリ360の記憶場所368、370、372、374及び376に記憶されている。以下にさらに詳細に説明するように、ディスク類似ディレクトリエントリ366は、図8及び図11の連係リストディレクトリエントリ158及び258とはそれぞれ異なっている。

【0192】図13は、ディレクトリエントリ366の構造の例を示す。図13は、ディレクトリエントリ366の一部であるディレクトリエントリ301及び361を示している。各ディレクトリエントリは固定長、たとえば40バイトである。

【0193】ルートディレクトリエントリ301は8バイトの名前311と、3バイトのファイル拡張313と、1バイトの属性フラグ315と、10バイトの予約

領域317と、2バイトの時間領域319と、2バイトの日付321と、2バイトの開始クラスタ323と、1バイトのクラスタ状態フラグ325と、4バイトのファイルサイズ領域327と、2バイトのポインタ329とを含む。

【0194】ファイル名311とファイル拡張313のバイトは、ルートディレクトリ301に対応するファイルを識別する。

【0195】属性フラグ315は8つのビット341、343、345、347、349、351、353及び355から構成されている。ビット341（ビット0）は、ファイルを読み取り専用ファイルとして、すなわち、ファイルの編集又は消去は不可能であるとしてマークする。ビット343（ビット1）は、ファイルがDOSに対して隠れており、DOSのディレクトリ指令及び他のいくつかのDOS指令によっては読み取りできないことを指示する。ビット345（ビット2）は、ファイルがディレクトリ指令及び他のいくつかのDOS指令によっては読み取りできないDOSファイルであることを指示するシステムビットである。

【0196】ビット347（ビット3）はボリュームラベルである。ビット349（ビット4）はサブディレクトリポインタである。ビット347又は349がセットされている場合には、DOSはそのエントリを通常のファイルディレクトリエントリとは異なる方法で処理しなければならない。

【0197】サブディレクトリビット349がセットされている場合には、ディレクトリエントリはサブディレクトリファイルであり、サブディレクトリファイルはフラッシュメモリアレイ302のブロックの中の1つに記憶されることになる。

【0198】ビット351（ビット5）は、ファイルがバックアップユーティリティによりバックアップされているか否かを指示するアーカイブビットである。ビット353（ビット6）とビット355（ビット7）は、使用されない予約ビットである。

【0199】予約バイト317は予約されており、使用されない。バイト319及び321はファイルに関する日時スタンプである。バイト327はファイルの大きさ、すなわち、ファイルの長さを指示する。

【0200】ルートディレクトリ301のバイト323及び325はルートディレクトリ301のファイル割付けテーブル（「FAT」）型ビットである。開始クラスタバイト323はフラッシュメモリアレイ302の中のファイルの開始アドレスである。バイト325はクラスタ状態バイトと呼ばれる。クラスタ状態バイト325は、バイト323に指示されている開始アドレスから始まるファイルの状態を指示する。状態バイト325は、ファイルが利用可能であるか、削除ファイルであるか又は欠陥ファイルであるかを指示する。

【0201】バイト323が参照する開始クラスタは模擬人工クラスタであることを理解すべきである。ファイルディレクトリエントリ301が参照するファイルは実際には可変長ファイルである。DOSが記憶される磁気ハードディスクドライブ（固定又はフロッピー）に見られるディレクトリ構造に幾分か類似しているディレクトリ構造を得るために、ルートディレクトリ301によってクラスタファイル構造をシミュレートする。

【0202】ファイルサイズバイト327に記憶されているファイルサイズが開始クラスタ323のサイズより大きければ、ファイルを完全に指定するために、複数の模擬クラスタを連鎖させなければならない。この動作は次のようにして実行される。ルートディレクトリ301のポインタバイト329は図13の付属ディレクトリエントリ361を指示する。付属ディレクトリエントリ361はファイル割付けテーブルバイト383及び385を含む。バイト383はクラスタバイトである。バイト385はクラスタ状態バイトである。

【0203】クラスタバイト383は、開始クラスタバイト323に記憶されているアドレスにより指示されるデータ又はコートと共に連鎖させるべきフラッシュメモリアレイ320内部のデータ又はコードの開始アドレスを記憶する。付属ディレクトリエントリ361のバイト385は、バイト383に記憶されているアドレスで始まるファイルの状態を指示する。クラスタ状態バイト385は、バイト383に記憶されているアドレスに記憶されたファイルが利用可能であるか、削除ファイルであるか又は欠陥ファイルであるかを指示する。

【0204】ファイルを作成するために他のデータ又はコードと共に連鎖すべきでない場合には、ディレクトリエントリ361のポインタ389は他のどのディレクトリエントリも指示しない。

【0205】ディレクトリエントリ361は単なる付属ディレクトリエントリであるので、名前バイト371、拡張バイト373、属性バイト375、予約バイト377、時間バイト379、日付バイト381及びファイルサイズバイト387の内容は問題にならない。

【0206】ディレクトリエントリの中には付属ファイルをもたないものもあるということを理解すべきである。さらに、他のいくつかのディレクトリエントリは複数の付属ファイルを有している。付属ファイルのポインタは別のファイルを指示することができる。

【0207】ファイル構造300は、均一な長さのディレクトリエントリを使用して、重ね書きを経てそれらを容易に再利用可能とすることにより、メモリ360のバイト可変性を利用している。また、エントリのサイズが対称であるために、エントリを従来のディスクのファイル割当てテーブルのエントリとして取扱うことができる。FAT型のクラスタ状態バイトを使用すると、ファイル構造300に関する状態報告は相対的に単純であ

り、DOSを記憶する従来のハードディスクドライブの状態報告に類似している。

【0208】メモリ360のペアレントディレクトリエントリは、ハード（又はフロッピー）ディスクがその別個のFATインデックスを使用してファイルを参照するように、組合せディレクトリ/FATインデックス記憶場所を参照する。

【0209】メモリ360内部の各ディレクトリ/FATエントリは、開始クラスタとしてファイルの開始アドレスを参照することにより、DOSを模倣している。ファイルの長さが模擬クラスタより長いときには、必ず、この第1のFATエントリに人工クラスタを連鎖させる。

【0210】メモリ360に記憶されているブートレコード362は、模擬クラスタサイズを記憶する。ブートレコード362は利用可能クラスタの数と、ディレクトリエントリの最大数とをさらに記憶する。利用可能クラスタの数はフラッシュメモリアレイ302の定様式化可能容量である。また、ブートレコード362は特定のファイルシステムの型及び特定のファイルシステムの改訂に関する情報を記憶している。ブートレコード362は、フラッシュメモリアレイ302に関する予約ブロック情報と、ディレクトリエントリ366におけるルートディレクトリの場所に関連する情報とを含む他のいくつかのファイルシステムパラメータをさらに記憶している。

【0211】パーソナルコンピュータシステムのオペレーティングシステムは、メモリ360内部のディレクトリ/FATエントリの数と、フォーマット上規定されている模擬クラスタサイズとを乗算した数と同等の数のセクターを含む。メモリ360内部のディレクトリ/FATエントリは、組合わせ領域368、370、372、374及び376のサイズをディレクトリエントリのサイズで除算した数に等しい。

【0212】図12のブロック状態テーブル364は内容の上では図8のブロック状態テーブル156と同様である。図12のブロック状態テーブル364は、フラッシュメモリアレイのどのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥ブロック又は故障ブロックであるかに関する情報を含む。

【0213】図12のフラッシュメモリアレイ302に関するクリーンアップ動作は、図8のフラッシュメモリアレイ112に関するクリーンアップ動作に類似している。図12のフラッシュメモリアレイ302のクリーンアップはブロック状態テーブル364と関連して進行し、先に図8のフラッシュメモリアレイ112に関連して説明した通り、循環方式で実行される。図12に示すファイル構造300の場合、ブロック316、318及び320は次にクリーンアップし且つ圧縮すべきブロックである。

【0214】パーソナルコンピュータシステムは、新たなエントリごとに次の利用可能なスペースがどこにあるかを知るために、ディレクトリと主フラッシュメモリアレイ302の利用状況を追跡しなければならない。ディスクに似たセクターが欠落しているならば、これは活動循環プロセスである。

【0215】ディレクトリエントリ366、ブートレコード362及びブロック状態テーブル364のサイズがメモリ360のサイズを越えて拡張することは不可能である。これに対し、サブディレクトリファイルはフラッシュメモリアレイ302の中に記憶される。

【0216】通常、フラッシュメモリの情報に重ね書きすることは不可能であるため、ディレクトリファイル366は一定数のエントリについてあらかじめスペースを割付けておき、次に、追加エントリのための付属を指示することが必要である。

【0217】図14は、フラッシュメモリアレイ401のファイル構造400を示す。ファイル構造は、複数のセクターと、動的フラッシュディレクトリとを含むファイル構造である。ファイル構造400をセクター方式ファイル構造ともいう。

【0218】ファイル構造400のようなセクター方式ファイル構造では、コード又はデータはセクター単位で記憶される。セクターはパーソナルコンピュータシステムの従来のハードディスクやディスクットにおけるセクターに類似している。従来のハードディスクの場合、トラックごとのセクター数は、通常、17である。ハードディスクによっては、クラスタごとに4つのセクターがあるものもある。他のハードディスクや高密度3.5インチディスクットでは、1つのクラスタは単一のセクターを含む。クラスタは論理的にアドレッシング可能な最小記憶単位である。

【0219】ファイル構造400の場合、1つのクラスタは単一のセクターから構成されている。各セクターは512バイト幅であり、これは固定長である。各クラスタを割付け単位（「AU」）ともいう。

【0220】本発明の別の実施例では、各クラスタは2つのセクターから構成されている。さらに別の実施例においては、各クラスタは3つ以上のセクターから構成されている。

【0221】ファイル構造400のセクター方式ファイル構造は、従来のディスクファイル構造のソフトウェア又はファームウェアドライバに基づくエミュレーションを単純化する助けになる。これは、ファイル構造400のセクター方式ファイル構造がハードディスクドライブに見られるセクター方式ファイル構造に類似しているためである。

【0222】また、ファイル構造400のセクター方式ファイル構造によって、クリーンアップに際して消去ブ

ロックをランダムに選択することが可能になる。このことが、今度は、ファイルの移動が不要であるときにそのような移動を少なくするのにも役立つのである。さらに、余分な循環を少なくするのにも役立つ。

【0223】その一方で、セクター方式ファイル構造はコードファイルの代わりとして完全連続実行を支援することができないのではあるが、連続セクターページングメモリファイル方式に従えば、大きな連続部分又はより小さなコードファイルをその場で実行させることが可能である。

【0224】セクター方式ファイル構造の場合、クリーンアップ中のホット/コールド消去ブロック再配分の活動管理を支援するために、個々のブロックの循環を追跡しなければならない。このような個々のブロックの循環は、本来、以下に論じるクリーンアップ/再割付け規則によって管理される。

【0225】フラッシュメモリアレイ401のブロック416、424及び428は「予約となるべき」ブロック（予約ブロック又は予約済ブロックともいう）である。ブロック408は、次のクリーンアップ動作の後に空きブロックになる空きとなるべきブロックである。

【0226】フラッシュメモリアレイ401のブロック404、414及び420は、コード及びデータを記憶するために使用可能なブロックである空きブロックである。ブロック430は、記憶場所464にはデータ又はコードを含み、記憶場所462には空きスペースを有する部分空きブロックである。

【0227】ブロック402、406、412、418及び426の中のセクターは、ブロック430の部分464にあるセクターと共に、アクティブファイル、削除ファイル、アクティブサブディレクトリ及び削除サブディレクトリを構成する。ブロック402の中のセクター403、405、409、411、413、415、417、423、425及び427と、ブロック406の中のセクター431、433、439、443、445及び453とはアクティブファイルとアクティブサブディレクトリを構成するセクターの例である。また、ブロック412、418、426、426及び430もアクティブファイルとアクティブサブディレクトリを構成するセクターを含む。セクター407、419、421、429、435、437、441、447、449、451、457、459、461、463、465、467、469、471、473、475、477、479、481、483、485、487、489及び491は、削除ファイルと削除サブディレクトリを含むセクターである。

【0228】ファイル割付けテーブル446の使用によって、ファイル構造400のクラスタは論理的に連結され、1つの完全ファイルを形成する。これは、固定でディスクドライブ又はフロッピーディスクドライブを伴う

従来のパーソナルコンピュータシステムで実行されていたのと同様である。すなわち、固定ディスク又はフロッピーディスクのクラスタも論理的に連結されて、1つの完全ファイルを形成する。

【0229】図14では、ファイル割付けテーブル446はブロック422の中に記憶されている。ファイル割付けテーブル446は、ファイル構造400が使用する動的ディレクトリ方式の一部である。ブロック422はディレクトリフラグ440と、ブートレコード442と、ブロック状態テーブル444と、FAT446と、関係リストディレクトリエントリ448と、空きスペース450とを含む。従って、ブロック422は現在ディレクトリを記憶している。

【0230】ディレクトリフラグ440は、ブロック422が関係リストディレクトリエントリ448を含むことを指示するデータのパターンを記憶する。

【0231】ブートレコード442はディレクトリフラグに続いている。ブートレコード442は特定のファイルシステムの型と、特定のファイルシステムの改訂とに関する情報を含む。ブートレコード442はいくつかのファイルシステムパラメータをさらに含む。それらのパラメータには定様式化可能容量情報と、予約ブロック情報と、付属ディレクトリ情報と、旧ディレクトリに関連する情報と、ルートディレクトリに関連する情報とがある。

【0232】ブロック状態テーブル444はブートレコード442に続いている。ブロック状態テーブル444は、どのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥又は故障ブロックであるかに関する情報を含む。ブロック状態テーブル444はブロックごとのサイクルカウント情報をさらに含む。このサイクルカウント情報は、1つのブロックを何度消去し、再書き込みしたかを指示する。

【0233】ファイル割付けテーブル446はブロック422のブロック状態テーブル444に続いている。ファイル割付けテーブル446は、関係リストディレクトリエントリ448の中に見出されるルートディレクトリの前に位置している。

【0234】関係リストディレクトリエントリ448は主ディレクトリ448とも呼ばれる。主ディレクトリ448は一連の個別ディレクトリエントリを含む。各ディレクトリエントリは（1）サブディレクトリを指示するポインタ（サブディレクトリはブロック422以外のブロックに記憶されている）又は（2）ファイルのディレクトリのいずれかである。図14のディレクトリエントリ448は、ファイル構造440の場合、物理主メモリ開始アドレスの代わりにFATポインタ446を使用している点を除いて、図8のディレクトリエントリ158に類似している。図14の各ディレクトリファイル

エントリはファイルの名前／拡張、日付、時間、ファイル型、ファイル長さ、シプリングポインタ及びチャイルドポインタに関する情報を含む。各ディレクトリファイルエントリは、1つの特定のファイルで使用する第1のクラスタの番号であるクラスタ番号をさらに含む。クラスタ番号は2バイトの大きさである。一実施例では、ファイル割付けテーブル446は、クラスタごとに1つずつの2バイトエントリから成るリストである。ファイル構造400のオペレーティングシステムは第1のクラスタの番号を使用して、ファイル割付けテーブル446をアクセスする。そのクラスタに関わるFATエントリは、ファイル中の次のクラスタのクラスタ番号を含む。FAT446の各エントリはファイルの次のクラスタに対するポインタである。オペレーティングシステムはこれを利用してファイルの次にクラスタをアクセスし、FAT446の特別なマーカに達するまで、このような動作を継続してゆく。本発明の一実施例では、その特別なマーカは16進数のFナルである。FAT446のエントリが0である場合には、これはクラスタが使用中でないことを指示する。

【0235】図14のファイル構造400の場合、FAT446の各FATエントリはフラッシュメモリアレイ401の物理クラスタに対して1対1の対応を有し、最後のファイル割付けクラスタに到達するまで連鎖するFATエントリを指示する。最終FATエントリは、依然として消去されている。すなわち、全て1のデータであるF（16進数）ナルデータにより指示される。

【0236】図14のファイル構造400のディレクトリエントリは、図14のファイル構造400の場合、物理主メモリ開始アドレスの代わりにFATポインタを使用している点を除いて、図8のファイル構造110のディレクトリエントリに類似している。図14のファイル構造400では、不連続のファイル情報をディスクに似たFAT連鎖によって結合する。

【0237】ファイル構造400が固定ディスク又はフロッピーディスクのファイル構造と異なる点の1つは、係リストディレクトリエントリ448の中でチャイルドポインタとシプリングポインタを使用していることである。

【0238】FATエントリと主メモリクラスタそれ自体は長さの上で対称であり、共に、短縮インデックス型アドレスから構成されている。FATエントリの最大数は一定であり、所定のクラスタサイズについての定様式化可能容量（すなわち、AUごとのセクター数）と、予約ブロックの数とを確定する。これは定様式化に際してブートレコード442に記録される。

【0239】ブロック状態テーブル444は、「空き」状態、「予約」状態及び「不良」状態に加えて、各ブロックについてのサイクルカウント情報を含む。ブロック状態テーブル444はディレクトリブロックと予約ブロッ

ックをアクセス不可能として予約して（後のクリーンアップを可能にする）、ファイルシステムがそれらのブロックに対し重ね書きを実行しようとする試みを阻止する。故障ブロックも同様に回避され、ブロック状態テーブル444の中に指示される。

【0240】クラスタは、一般に、ある1つの空きブロックの終端を越えて連続して書込まれてゆき、別の空きのブロックに書込まれる後続クラスタに連結される。この通常連続しているクラスタ使用法は、適切に指定されたコードファイルについてページング方式現場実行動作を支援することができる。後続するファイル削除は主フラッシュメモリアレイの全体にランダムな未使用部分を発生させる。

【0241】クリーンアップが要求されると、パーソナルコンピュータシステムは後の再書込みに備えてどのブロックをクリアすべきかを識別する。言い換えれば、クリーンアップ時、コンピュータシステムは、クリーンアップに際して再割付けし、移動させるべきファイルを有するブロックを識別する。クリーンアップ時にクリアすべきブロックの識別は2つの規則により管理される。第1の規則は、循環の最も少ないブロックを選択するというものである。これはホットブロックとコールドブロックとの循環の不均衡を減少させる。

【0242】ブロックのサイクルカウントに大きな矛盾が存在しない場合には、第2の規則が管理する。言い換えれば、循環の配分がコンピュータシステムにより問題としてフラグ付けされない場合には、第2の規則が管理する。第2の規則は、最大数の削除セクターを伴うブロックがクリアのために選択されるブロックであると規定する。

【0243】この第2の規則の根拠は、削除ファイルを含むクラスタを消去に先立ってブロックから取出す必要はないということである。削除ファイルはセーブしなければならない情報を全く含んでいないので、削除ファイルをクリーンアップ動作の一部として消去することはできない。言い換えれば、削除ファイルの再割付けの必要はないのである。従って、第2の規則を採用すると、クリーンアップ動作の一部である消去に先立ってアクティブセクター、すなわち、アクティブクラスタから取出す量は最小限に抑えられる結果となる。

【0244】どのブロックが最多の削除ファイルクラスタを含むかを管理するのは、本質的にはランダムなプロセスである。従って、第2の規則は、クリーンアップ方式によって、クリアすべきブロックに関してブロックの選択の大部分がランダムになるということを意味する。この本質的にランダムなブロック選択はクリーンアップ効率を向上させるのを助けると同時に、総体的な循環の不一致を最小に抑える。

【0245】上記の第1と第2の規則が合わせてフラッシュメモリアレイ401内部の循環配分を管理する。こ

れら2つの規則は合わせて循環の差異を最小限に抑えるのを助ける。従って、ファイル構造400のクリーンアップは本質的にランダムである。

【0246】図14のファイル構造400の場合、クリーンアップ時に新たなディレクトリ（及び新たな付属ディレクトリ）は新たなFATと共に予約ブロックの中に構成される。

【0247】一実施例では、ファイル構造400はフラッシュメモリアレイ401に記憶されているヘッダを含む。ヘッダは図8のファイル構造110に関して論じたヘッダに含まれている情報に類似する情報を含む。

【0248】図15は、フラッシュメモリアレイ501のファイル構造500を示す。ファイル構造500は複数のセクターを含むファイル構造であり、セクター方式ファイル構造と呼ばれる。

【0249】セクター方式ファイル構造500では、データ又はコードはセクター単位で記憶される。図15の実施例では、クラスタごとに1つのセクターがあり、各セクターは512バイトである。

【0250】ブロック508、516及び524はファイル構造500の予約ブロックである。ブロック508は空きとなるべきブロックである。

【0251】ブロック504、514及び520は空きブロックである。ブロック530は記憶場所564にデータ又はコードを記憶し、記憶場所562には空きスペースを有する一部空きブロックである。

【0252】ブロック502、506、512、518及び526の中のセクターは、ブロック530の部分564の中のセクターと共に、アクティブファイル及び削除ファイルと、アクティブサブディレクトリ及び削除サブディレクトリとを構成する。ブロック502の中のセクター503、505、509、511、513、515、517、523、525及び527と、ブロック506の中のセクター531、533、539、543、545及び553はアクティブファイルセクターと、アクティブサブディレクトリセクターの例である。ブロック512、518、526及び530もアクティブファイルセクター及びアクティブサブディレクトリセクターを含む。セクター507、519、521、529、535、537、541、547、549、551、557、559、561、563、565、567、569、571、573、575、577、579、581、583、585、587、589及び591は、削除ファイルと削除サブディレクトリを含むセクターである。

【0253】図15に示すファイル構造500の場合、ブートレコード542と、ブロック状態テーブル544と、ファイル割付けテーブル546と、連係リストディレクトリエントリ548と、空きディレクトリスペース550とは別個のメモリ534に記憶されている。ただ

し、別個のメモリ534は、フラッシュメモリアレイ501を含むパーソナルコンピュータシステムの一部として含まれている。

【0254】本発明の一実施例においては、メモリは従来のバイト可変EEPROMである。

【0255】別の実施例では、別個のメモリ534はバックアップとしてのバッテリーに結合するSRAMである。

【0256】さらに別の実施例では、別個のメモリ534はNVRAMである。図15に示す実施例においては、メモリ534はEEPROMである。定義上、連係リストディレクトリエントリ548がメモリ534の中に見出されるとすれば、EEPROM534にディレクトリフラグを記憶する必要はない。言い換えれば、メモリ534には現在ディレクトリが記憶されている。

【0257】一実施例では、ファイル構造500はフラッシュメモリアレイ501の中に記憶されているヘッダを含む。ヘッダは図8に関して論じたヘッダに含まれている情報と類似する情報を記憶する。

【0258】図15のファイル構造500は、別個のメモリ534の固定記憶場所にディレクトリエントリ548を有するという点を除いて、図14のファイル構造400に類似している。すなわち、ファイル構造500はメモリ534に関してディレクトリフラグを有しておらず、最大数のディレクトリエントリが存在しているのである。

【0259】ランダムな連係リストディレクトリ修正を簡略化するために、ディレクトリエントリ548も同じように長さの上で対称である。ランダムな、相対的に効率の良いブロック再割付けの間には、循環管理のためにブロック状態テーブル544を使用する。他の全てのディレクトリ/FATの動作は図14のファイル構造400の場合の動作に類似している。

【0260】図16は、フラッシュメモリアレイ601のファイル構造600を示す。ファイル構造600は複数のセクターを含むファイル構造であり、セクター方式ファイル構造と呼ばれる。

【0261】セクター方式ファイル構造600では、データ又はコードはセクター単位で記憶される。図16の実施例では、クラスタごとに1つのセクターがあり、各セクターは512バイトである。ブロック608、616及び624はファイル構造600の予約ブロックである。ブロック608は空きとなるべきブロックである。ブロック604、614及び620は空きブロックである。ブロック630は記憶場所664にデータ又はコードを含み、記憶場所662には空きスペースを有する一部空きブロックである。

【0262】ブロック602、606、612、618及び626の中のセクターは、ブロック630の部分664にあるセクターと共に、アクティブファイル及び削

除ファイルと、アクティブサブディレクトリ及び削除サブディレクトリとを構成する。ブロック602の中のセクター603, 605, 609, 611, 613, 615, 617, 623, 625及び627と、ブロック606の中のセクター631, 633, 639, 643, 645及び653とはアクティブファイルセクターとアクティブサブディレクトリセクターの例である。ブロック612, 618, 626及び630もアクティブファイルセクターと、アクティブサブディレクトリセクターを含む。セクター607, 619, 621, 629, 635, 637, 641, 647, 649, 651, 657, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 689及び691は、削除ファイルと削除サブディレクトリを含むセクターである。

【0263】図16に示すファイル構造600の場合、ブート記録642と、ブロック状態テーブル644と、ファイル割付けテーブル646と、ディレクトリエントリ648とは、フラッシュメモリアレイ601の一部ではない別個のメモリ634に記憶されている。ただし、この別個のメモリ634は、フラッシュメモリアレイ601をも含むパーソナルコンピュータシステムの一部として含まれている。

【0264】本発明の一実施例においては、別個のメモリ634は従来のバイト可変EEPROMである。別の実施例では、別個のメモリ634はバックアップとしてのバッテリーに結合するSRAMである。さらに別の実施例ではメモリ634はNVRAMである。

【0265】図16に示す実施例においては、メモリ634はEEPROMである。定義上、メモリ634にディレクトリエントリ648が見出されるのであれば、EEPROM634にディレクトリフラグを記憶する必要はない。言い換えれば、メモリ634には現在ディレクトリが記憶されているのである。

【0266】一実施例では、ファイル構造600はフラッシュメモリアレイ601の中に記憶されているヘッダを含む。このヘッダは、図8に関して論じたヘッダに含まれている情報に類似する情報を記憶する。

【0267】図16のファイル構造600は、(1) パーソナルコンピュータシステムにおけるハードディスク又はディスクットのディレクトリ構造に類似するディレクトリ構造648を有することと、(2) 別個のメモリ634にディレクトリエントリ648を記憶していることを除いて、図14のファイル構造400に類似している。ファイル構造600はメモリ634に関してディレクトリフラグを有していない。

【0268】図16のファイル構造600は、このように、ディスク類似ディレクトリ構造を有しているのである。この実現形態は、単純に、磁気ディスクのディレク

トリをFATの規約に従っている。バイト可変性があるため、ディレクトリ及びFATのセクター利用/連鎖情報を直接に重ね書きすることが可能である。一次フラッシュ特有アスペクトは、消去ブロックレベルでまず消去して始めて論理セクターを重ね書きができるということになっている。従って、クリーンアップは図8のファイル構造110の規約に従うのであるが、サブディレクトリファイルエントリ方式と、他の大半の制約や特性は図12のファイル構造300に類似している。図16のファイル構造600は従来の磁気ディスクの動作に類似する方式で動作し、ディスククラスタに類似するファイルやサブディレクトリのクラスタの読取り及び書込みを実行する。ブロック状態テーブル644の利用とクリーンアップ動作は重要なフラッシュ特有の要素である。好ましい実施例においては、従来のディスクでは典型的である「バックアップディレクトリ/FAT」方式を採用する。

【0269】本発明の好ましい実施例は、固有のフラッシュ循環管理によって装置の信頼性の向上を促進する。また、本発明の好ましい実施例は、固有のバックアップコピー作成、すなわち、アクティブファイル及び/又はディレクトリ情報の重ね書きの回避によって、ファイルシステムの信頼性の向上を促進する。

【0270】本発明の好ましい実施例は(1) ファームウェアに制御コードを埋込み且つ(2) クリーンアップ中の直接ブロック間ファイル/ディレクトリ転送のために予約ブロックを使用することにより、システムRAMに課される要件をできる限り緩和する。

【0271】本発明の好ましい実施例は、連係リストディレクトリ構造の使用によって、フラッシュメモリスペースの浪費を最小限に抑える。

【0272】本発明の可変ファイルの実施例によれば、フラッシュレイの小さな管理可能部分のクリーンアップを経て再割付けが可能である。

【0273】セクター方式/動的ディレクトリを採用する実施例は、ブロック状態テーブルを使用するのであれば、性能、信頼性及び効率の向上を促進する。ブロック状態テーブルは(1) ホットスポットが他の装置及び装置内のブロック(それらが存在する場合)に対して消去及び書込み性能の劣化を発生させるのを阻止する一方で、装置の信頼性をできる限る高めるための均一なフラッシュ装置とブロックの循環、並びに(2) 信頼しうる急速なクリーンアップ及び「バックグラウンドタスク」消去のための予約ブロックの追跡を支援する。

【0274】以上の明細書においては、本発明をその特定の実施例に関連させて説明したが、特許請求の範囲に記載したような本発明のより広い範囲の趣旨から逸脱せずに様々な変形や変更を実施しうることは自明であろう。従って、明細書と図面は限定的な意味ではなく、例示を目的とするものとしてみなされるべきである。

【図面の簡単な説明】

【図1】従来のオペレーティングシステムの論理的編成を示す図。

【図2】フラッシュアレイと、コンピュータシステムのRAMに記憶されているファイルシステムドライバとを含むパーソナルコンピュータシステムを示す図。

【図3】 (1) フラッシュメモリアレイと、(2) ROM BIOS及びROMBIOS拡張部分の中に含まれているファイルシステムドライバと、(3) ファイルを転送し且つファイルのディレクトリを更新するRAMバッファとを含むパーソナルコンピュータシステムを示す図。

【図４】（１）フラッシュメモリアレイと、（２）ROM BIOS及びROM BIOS拡張部分に含まれているファイルシステムドライバと、（３）別個のシステム制御装置とを含むパーソナルコンピュータシステムを示す図。

【図5】入出力マッピング方式のフラッシュメモリアレイを示す図。

【図6】 ページング形マッピング方式のフラッシュメモリレイ構造を示す図。

【図7】直接メモリマッピング方式のフラッシュメモリアレイ構造を示す図。

【図8】可変ファイル構造及び動的ディレクトリを有するフラッシュEEPROMを示す図。

【図9】 連係リストディレクトリの1例の論理的配列を

示す図。

【図10】連係リストディレクトリの1例の論理的場所を示す図。

【図 11】可変ファイル構造と、ディレクトリを記憶する別個のEEPROMとを有するフラッシュEEPROMを示す図。

【図１２】可変ファイル構造と、ディスク類似ディレクトリを記憶する別個のＥＥＰＲＯＭとを有するフラッシュＥＥＰＲＯＭを示す図。

【図13】2つのディレクトリ/FATエントリを示す図。

【図14】セクター方式ファイル構造及び動的ディレク
トリを有するフラッシュEEPROMを示す図。

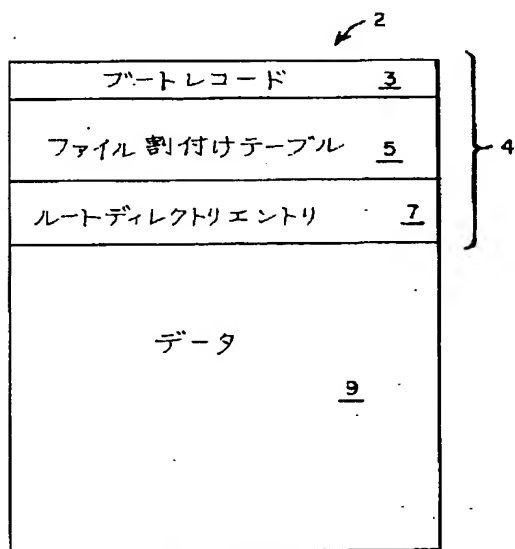
【図15】セクター方式ファイル構造と、ディレクトリを記憶する別個のEEPROMとを有するフラッシュEEPROMを示す図。

【図16】セクター方式ファイル構造と、ディスク類似ディレクトリを記憶する別個のEEPROMとを有するフラッシュEEPROMを示す図。

【符号の説明】

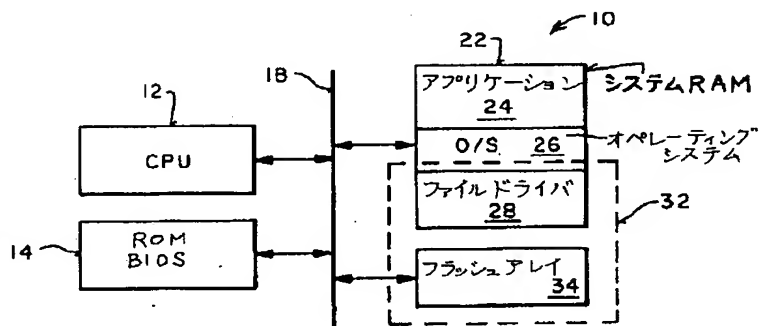
- 10 パーソナルコンピュータシステム
12 中央処理装置
32, 58, 94 ファイル構造
34, 64, 96 フラッシュメモリアレイ
92 制御装置

【図 1】

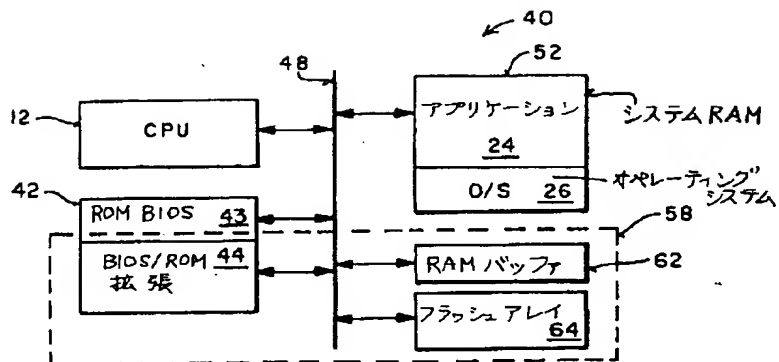


(従来の技術)

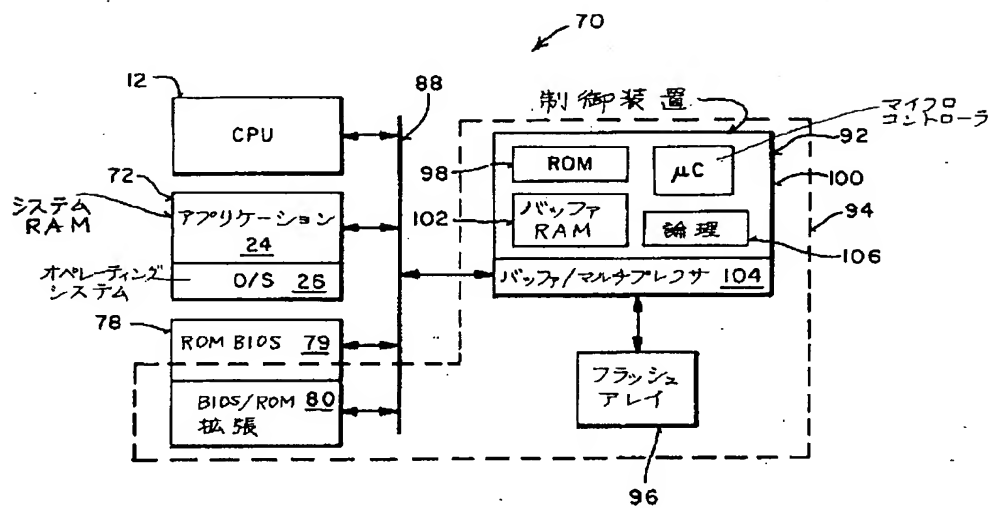
【图 2】



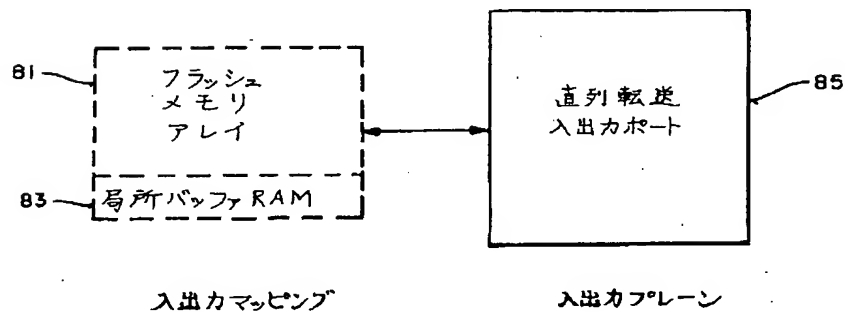
【図3】



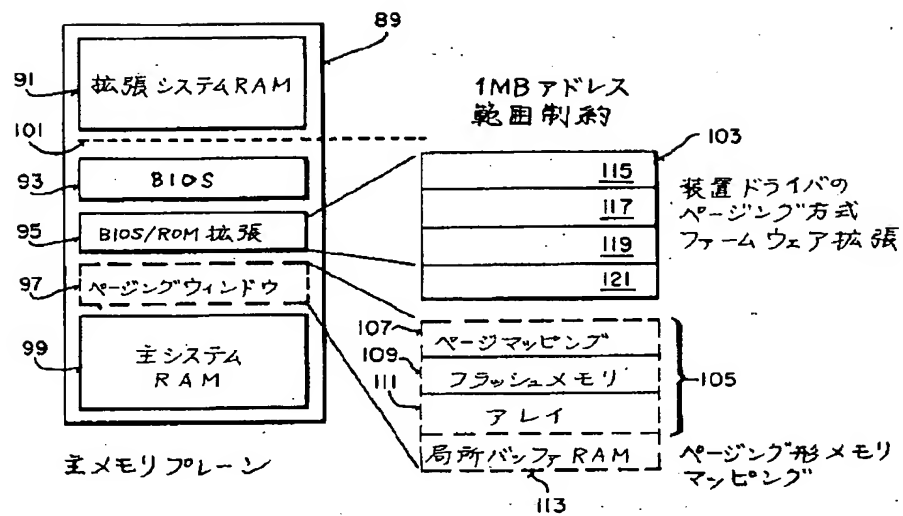
【図4】



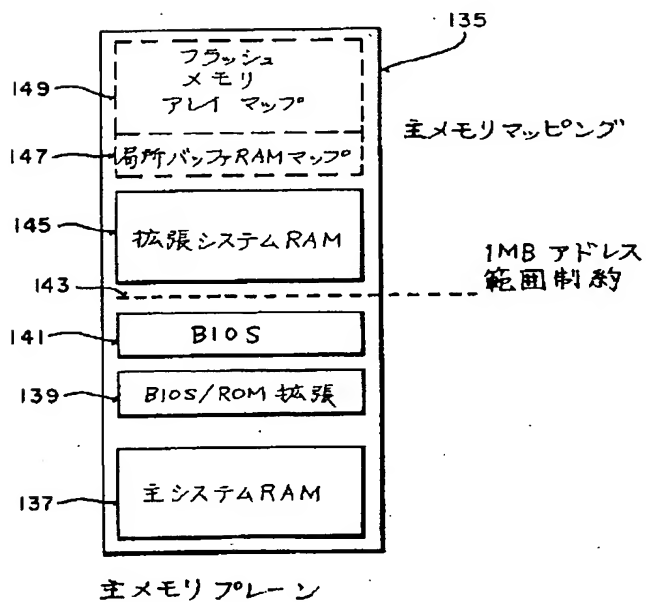
【図5】



【図6】

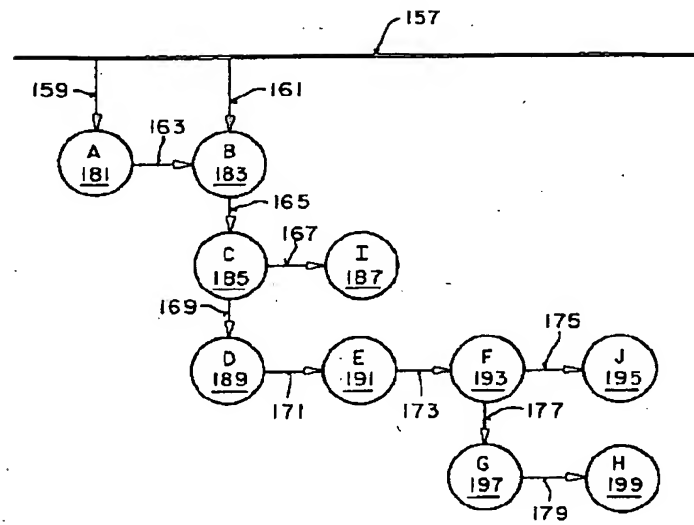


【図7】

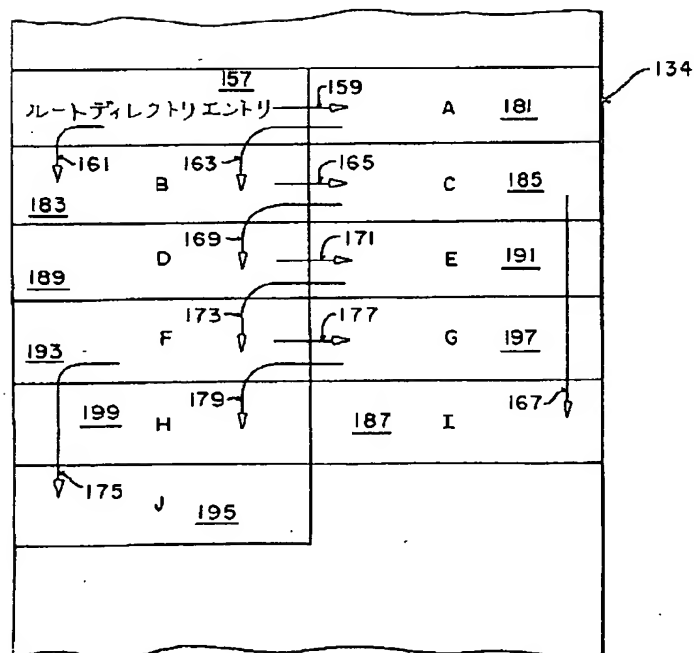


[illegible]

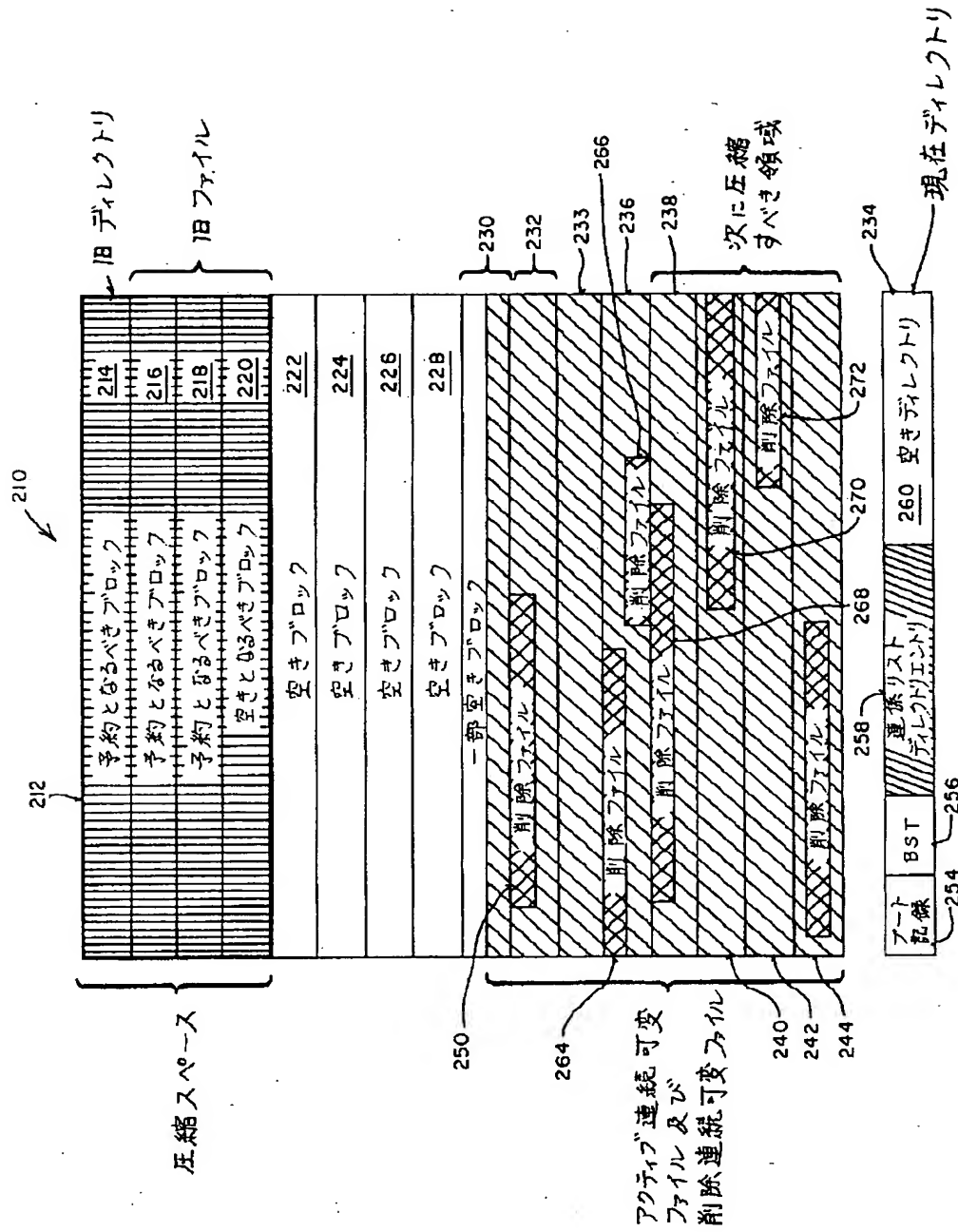
【図9】



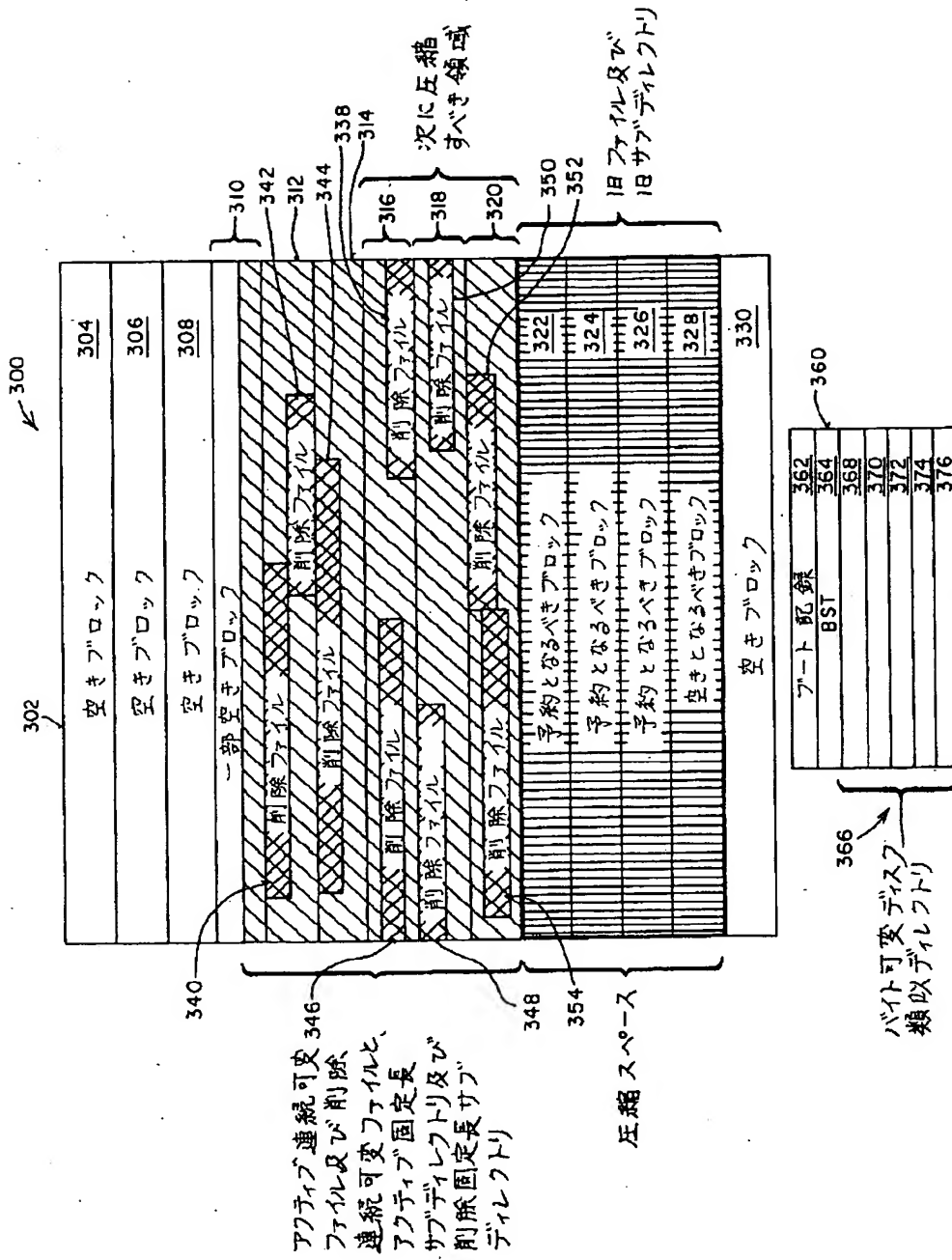
【図10】



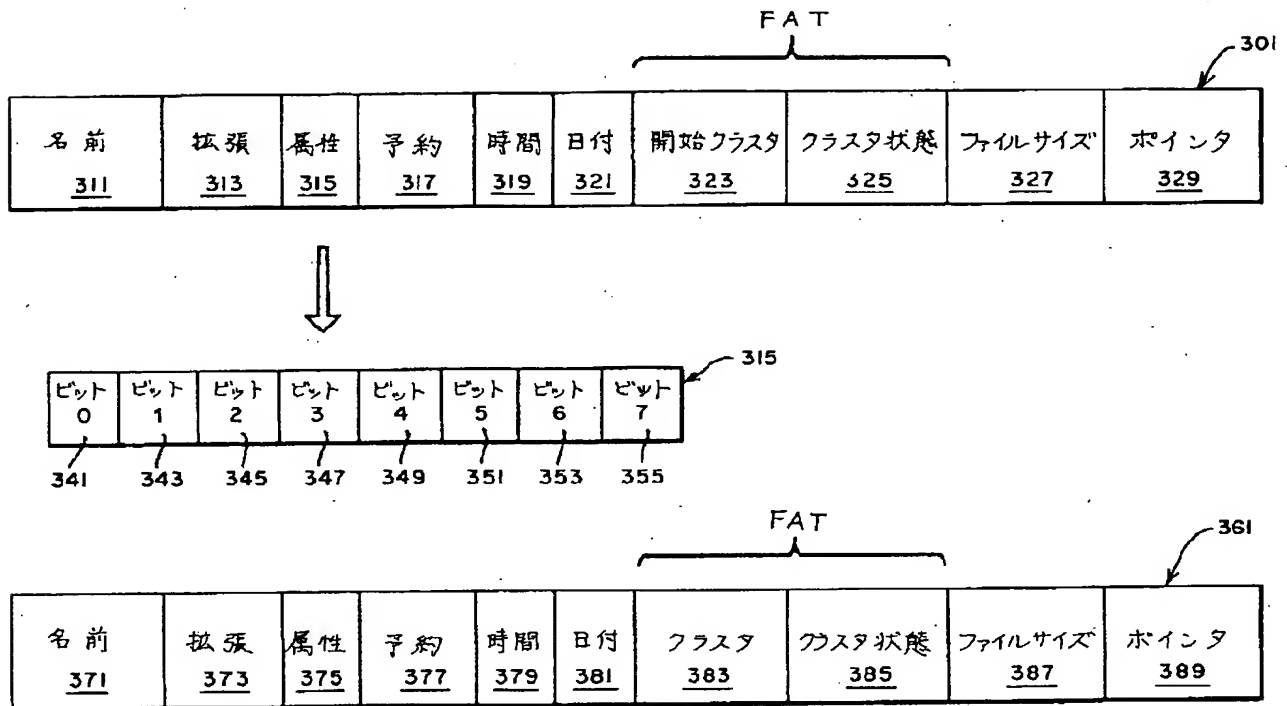
【図 1 1】



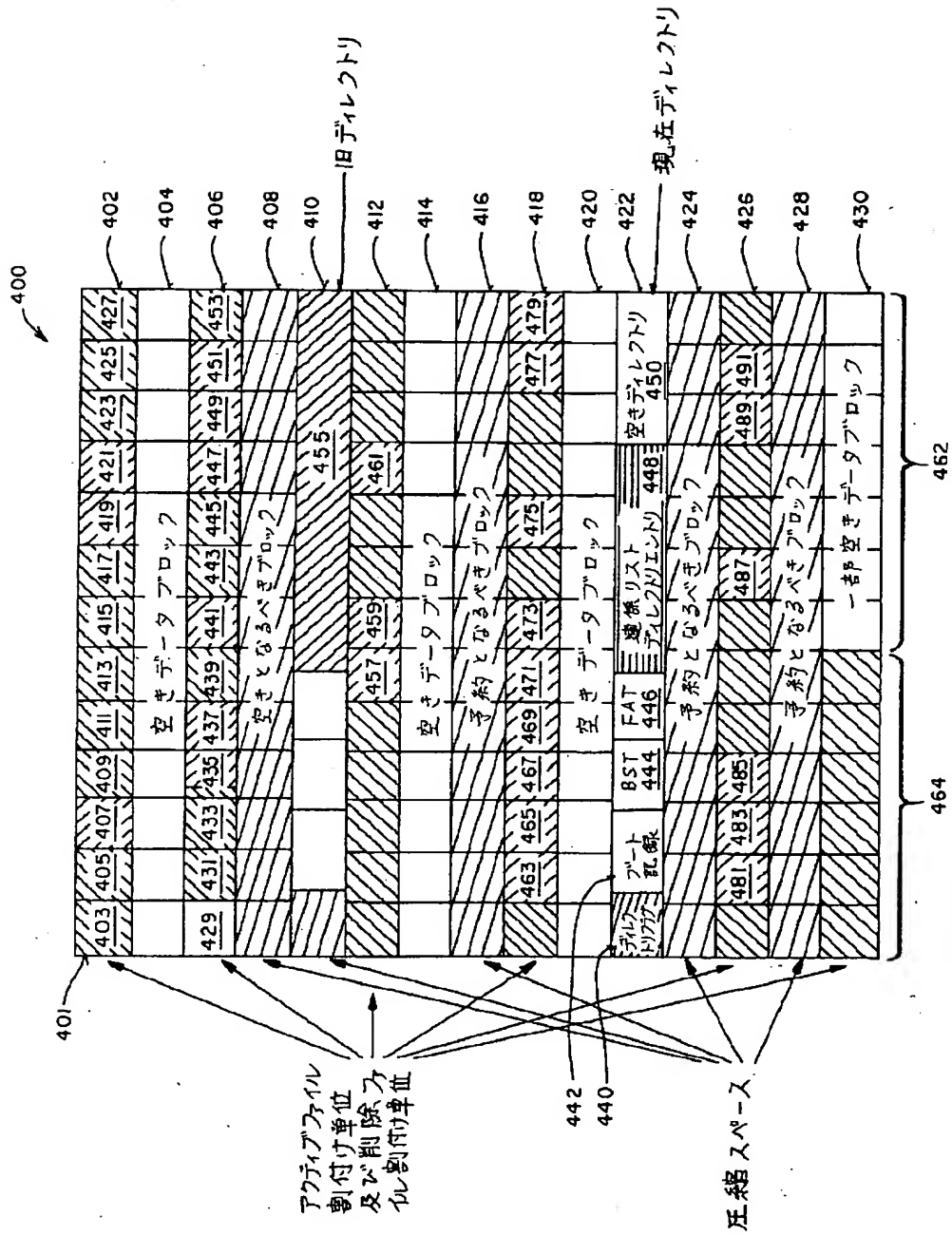
【図12】



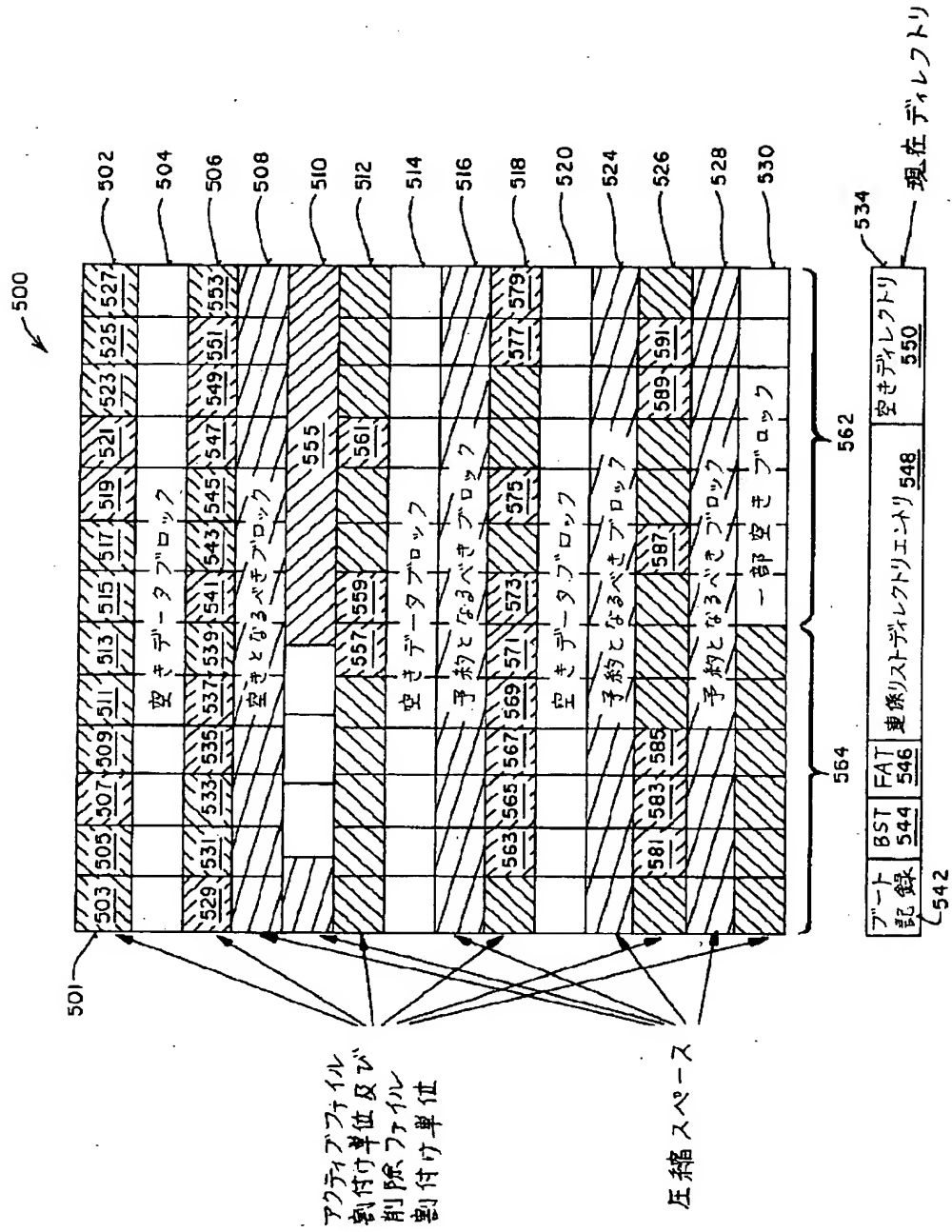
【図13】



【図 14】



【図15】



【図16】

